Modeliranje vremenskih nizova pomoću TensorFlow STS

Samardžić, Marko

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:528202

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-14**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

ZAVRŠNI RAD

Modeliranje vremenskih nizova pomoću TensorFlow Structural Time Series Paketa

Marko Samardžić

Pula, Rujan 2019.

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

ZAVRŠNI RAD
Modeliranje vremenskih nizova pomoću TensorFlow Structural Time Series Paketa

Marko Samardžić

JMBAG:

Kolegij: Modeliranje i simulacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi. informatologija

Mentor: doc. dr. sc Darko Etinger

Komentor: dr. sc. Nikola Tanković

Pula, Rujan 2019

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA
o korištenju autorskog djela

Ja, _____ **dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom** _____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

Zahvaljujem se svima koji su mi omogućili i pomogli pri izradi završnog rada. Posebno zahvaljujem komentoru dr. sc. Nikoli Tankoviću na savjetima, vodstvu i prenesenom znanju. Veliko hvala mojoj obitelji, prijateljima i djevojci Heleni, koji su mi bili velika podrška i motivacija.

SADRŽAJ

1. UVOD....1
   1.2 Karakteristike vremenskih nizova....1
   1.3 Rizici prognoziranja.... 2
2. Priprema i vizualizacija podataka.... 3
3. Model eksponencijalnog izglađivanja....6
   3.2 Model jednostavnog prosjeka 7
   3.3 Model pomičnog prosjeka....9
   3.4 Model jednostavnog eksponencijalnog izglađivana 10
   3.5 Holt'sova Linearna Trend metoda....12
   3.6 Holt-Winter model....13
4. ARIMA model....14
   4.2 Stacionarnost i nestacionarnost vremenskih nizova....16
   4.5. Dekompozicija....23
   4.6 Identifikacija modela i pronalažanje optimalnih parametara....25
   4.7. Prediktivni model....28
5. Modeliranje pomoću TensorFlow Structural Time Series bibilioteke....32
   5.2 Strukturalni vrmenski nizovi....33
   5.3 Local level model....33
   5.4 Local linear trend model....34
   5.5 Local linear trend model uz sezonalnost....36
   5.6 Strojno učenje i vjerojatnost....37
   5.7 Kalman Filter....39
   5.8. Varijacijska razlika....40
   5.9 Kako mjeriti sličnost dviju distribucija?....42
6. Zaključak....45

1.Uvod

S porastom tehnologije u današnjem svijetu pojavljuje se pojam "*Big data*". Razni podaci zabilježeni koristeći internet i softver rapidno rastu. Vrlo se brzo došlo do ideje da upravo ti podaci ne stoje samo u bazi podataka te da služe samo u svrhu jednostavnog prikazivanja, uređivanja, brisanja i ostalih funkcija, već da se iz njih proba dobiti neko konkretno znanje. Informacije koje mogu proizaći iz tih podataka mogu biti jako bitne i korisne za poslovanje i svijet općenito. Vrlo često se može do zaključaka doći i jednostavnim statističkim metodama. U drugim slučajevima tu uskaču i algoritmi strojnog učenja te napredna analitika. Kao što možemo promatrati povijesne podatke i donositi zaključke, tako možemo te podatke koristiti za previđanje budućnosti.

Kada se govori o vremenskim nizovima, upravo ta perspektiva u budućnost je ključ analize. Promatrajući vremenske nizove nastojimo predvidjeti kretanje promatrane varijable. Primjene su mnoge, od globalnog zatopljena, emisija ugljikovog dioksida, predsjedničkih izbora, financijskih izvještaja itd. Uzevši to u obzir možemo pretpostaviti da prognoza vremenskih nizova može biti jako važna. Predikcija može rezultirati određenim preventivnim mjerama, može biti informativna, služiti kao nekakav okvir za pouzdanost u odluci i slično.

U ovom radu obradit će se nekolicina metodologija, pristupa i tehnika u svrhu što bolje prognoze vremenskih nizova. Budući da se radi s podacima koji imaju svoje karakteristike, ne možemo garantirati najbolju metodu za svaki slučaj. Proći ćemo kroz vrste vremenskih nizova i čimbenike koje takav niz obilježuju. Upravo takve stvari ponekad uvjetuju metodu i pristup. Postoji naravno i neke stvari na koje treba obratiti pozornost prilikom svakog modeliranja vremenskih nizova, a neovisni su o odabiru metode. Cilj ovog rada je prikazati takve i sve ostale tehnike vezane za vremenski niz te na koncu izgraditi što bolji prediktivni modeli. Prikazat ćemo izradu modela od samog početka gdje se radi čišćenje i pripremanje podataka do završnih metoda poput procjene točnosti i odabira optimalnog prediktivnog modela. Prediktivni modeli se bave estimacijom određene varijable u budućnost. Kod svake metode bit će pojašnjena i njena intuicija i način na koji radi.

1.2 KARAKTERISTIKE VREMENSKIH NIZOVA:

Vremenski nizovi mogu se opisati kao skup podataka jednake pojave ili opservacije prikupljene unutar nekog vremenskog intervala. Tom opisu praktički može odgovarati skoro svaki podatak, stoga svaki skup podataka može biti promatran kroz vremenski niz. Međutim ta perspektiva koju vremenski niz pruža nije uvijek potrebna. Ako nema neke bitne vrijednosti u promatranju određene varijable kroz vrijeme i definiranja vremenskih intervala kroz tu varijablu onda možemo odbaciti takvu perspektivu.

Predviđanje i prognoza može imati razne oblike — kristalna kugla, čitanje iz kave, sportska kladionica, *brainstorming*, generiranje scenarija, što-ako analiza, razne statističke simulacije slučaja poput *Monte Carlo* simulacije. [1]

Tema ovog rada jest statističko predviđanje vremenskih nizova. Koristeći se znanstvenim tehnikama i metodama u svrhu predviđanja podataka u budućnost na temelju onoga što do sada znamo ili ne znamo. Ovakva metoda je vrlo elokventan način za opisati i istaknuti informacije koje su ovisne o vremenu kroz koje promatramo podatke. Modelirajući i analizirajući vremenski niz možemo donijeti intuitivne i značajne odluke za buduće poslovanje, predviđanje i slične probleme odnosno rješenje.

Ideja je u skupu pronaći statistički relevantne uzorke koji se pojavljuju zajedno s svojim pravilima i karakteristikama te ako smo dovoljno pouzdani da će se takvi uzorci ponoviti u budućnosti; uzeti ih u obzir pri izgradnji modela.

Iako na prvu može zvučati poprilično jasno i direktno kako krenuti rješavati takav problem, u većini slučajeva potrebno je imati jako izražene analitičke sposobnosti, znanja, iskustva i dobro poznavanje promatranog skupa podatka. Jako su bitni i kutevi gledanja na podatke te što je sve uzeto u obzir kada su se donosili zaključci.

Lako je za zaključiti da model može biti loš ako se pri njegovoj izradi nije uzelo sve u obzir o podacima i nešto se propustilo napraviti ili proučiti. Upravo iz tog razloga potrebno je osigurati da su se podaci pogledali iz više kuteva i na više načina prije samog razvijanja modela.

Karakteristike koje su usko vezane uz vremenski niz su pojmovi poput sezonalnosti, trenda i cikličnosti.

Prije izrade i upoznavanja s osnovnim metodama modeliranja vremenskih nizova predstavit ćemo pojmove koji vremenski niz obilježavaju.

Trend postoji u vremenskom nizu onda kada promatrani podaci imaju tendenciju rasta ili pada kroz vrijeme. Primjerice, podaci koji su u svojoj distribuciji linearni nemaju trend, dok podaci koji imaju određeni smjer kroz vrijeme sadrže komponentu trenda.

Sezonalnost je uzorak u podacima kada je promatran skup pod utjecajem sezonskih faktora kao što su doba godine, dio dana ili tjedna. Sezonalnošću se smatra uzorak koji je konzistentan u svom ponavljanju.

Cikličnost se pojavljuje kada distribucija vremenskog niza mijenja smjer bez fiksne frekvencije. [2] Tu se u suštini razlikuje od sezonalnosti. Cikličnost se definira kao neponavljajuća pojava.

Razumijevanje objašnjenih pojava ključno je u izgradnji dobrog modela te shvaćanju metoda koje se koriste za vremenske nizove. Svaki dan skup sa sobom nosi pojedine karakteristike, te je od velike važnosti uzeti sve u obzir.

1. 3 Rizici prognoziranja

Predviđanje ima svoje rizike i neizvjesnosti. Postoje metode i mjere koje rizik pokušavaju smanjiti.

Unutarnji rizik - je slučajna varijacija u nizu koju ne možemo objasniti s našim podacima i alatima koje imamo. Možemo to nazvati kao "buka" u sustavu. Ona se obično mjeri kao standardna pogreška modela. Gotovo u svakom sustavu barem u nekoj mjeri takav rizik je prisutan. Takav rizik možemo smanjiti da probamo naći uzorke i objašnjenja koja su prije smatrana bukom i slučajnosti. Te kada to novo promatranje uzmemo u obzir dolazimo do novih, točnijih prognoza.[1]

Rizik parametra - rizik koji se manifestira kroz pogreške u estimiranju parametara za prediktivni model koji koristimo. Mjeri se kao standardna pogreška procjene nagiba u trend liniji. Uzimajući što veći broj podataka koji promatramo ovakvu pogrešku možemo smanjiti. No treba biti oprezan jer u vremenskim nizovima nerijetko više podataka ne znači bolje, naročito jer su nizovi specifični zbog svoje vremenske komponente. Znamo da niti jedan uzorak i pojava ne ostaju isti kroz jako dugo vrijeme. [1]

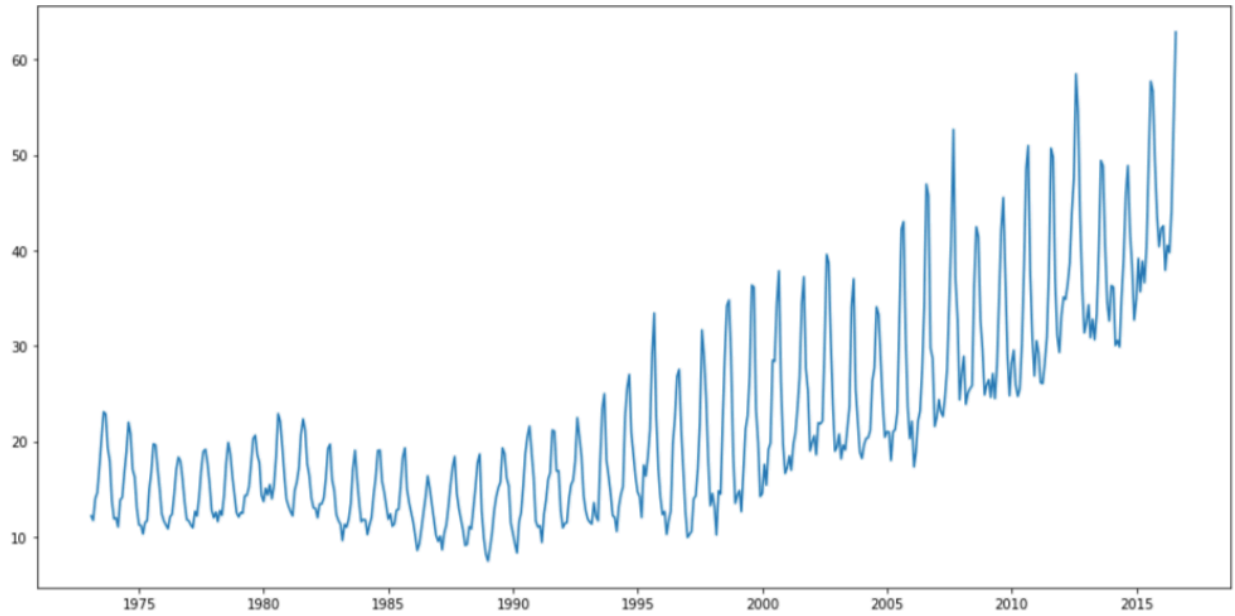
Rizik modela - misli se na izbor pogrešnog modela. Najčešće dolazi zbog krive procjene o podacima itekako će se budućnost propagirati. Ovo je najozbiljniji i najveći rizik i pogreška koju možemo napraviti u prognoziranju. Uz to je i naj izazovniji problem jer nema standardne pogreške ili slične mjere kako bi s brojkama potkrijepili našu odluku zato što svaki model po sebi pretpostavlja da je točan. Praćenje dobrih statističkih praksi i pristupa mogu pomoći u smanjenju krivog odabira modela. Najbitnije je razumjeti i upoznati svoje podatke te uzeti u obzir različite kuteve gledanja i ne slijepo se osloniti na točnost jednog ili više parametara koji ipak možda nisu vjerni pokazatelj ispravnog modela. [1]

U sljedećim primjerima i tehnikama proći ćemo kroz cijeli postupak izgradnje, analize i vizualizacije vremenskog niza kao skupa podataka. Radit ćemo razne testove kako bi što bolje upoznali prirodu skupa i kakve su njegove karakteristike, te sukladno njima odlučiti koji model je najpogodniji za izgradnju prediktivnog modela.

2. Priprema podataka i vizualizacija podataka

Kao i uvijek, najprije ćemo prikazati kako izgledaju naši podaci te ih vizualizirati kako bi dobili neku generalnu polaznu sliku.

Za analizu koristim javno dostupan skup podataka o mjesečnom ispuštanju ugljikovog dioksida.



Slika 1. Emisija ugljikovog dioksida, izvo podataka: http://www.eia.gov/electricity/data.cfm#elecenv

Naredbom `data.info()` možemo vidjeti kakvi su naši podaci u smislu tipa (cjelobrojni, objekt, float itd) te koliko ih ima. Vidimo da imamo 5093 instanci odnosno redaka koje promatramo.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5094 entries, 0 to 5093
Data columns (total 6 columns):
MSN             5094 non-null object
YYYYMM          5094 non-null int64
Value           5094 non-null object
Column_Order    5094 non-null int64
Description     5094 non-null object
Unit            5094 non-null object
dtypes: int64(2), object(4)
memory usage: 238.9+ KB
```

Slika 2. Opis promatranog skupa. Izvor: vlastiti rad

Prije početka same manipulacije i analize nad podacima, moramo se pobrinuti za format određenih podataka u skupu. Tu se najviše pažnje skreće na datume zato što mogu biti u raznim oblicima.

Prilagodit ćemo kolonu 'YYYYMM' formatu koji nam je potreban kako bi mogli kroz Pandas i druge pakete raditi nad podacima.

```
dateparse = lambda x: pd.to_datetime(x, format = '%Y%m', errors = 'coerce')
data = pd.read_csv(url, parse_dates = ['YYYYMM'], index_col='YYYYMM', date_parser =
dateparse)
data.head()
```

YYYYMM	MSN	Value	Column_Order	Description	Unit
1973-01-01	CLEIEUS	72.076	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-02-01	CLEIEUS	64.442	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-03-01	CLEIEUS	64.084	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-04-01	CLEIEUS	60.842	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-05-01	CLEIEUS	61.798	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide

Slika 3. Prikaz podatka. Izvor: vlastiti rad

Vidimo u tablici da su datumi sada drugačije formatirani te je promjena evidentirana.

Pojašnjenje argumenata:

- `parse_dates` - spremamo i identificiramo kolonu koja sadrži vrijeme odnosno datume vremenskog niza
- `index_col` - naredba kojoj deklariramo index kolonu u Pandas *data frameu*. Index kolona treba biti date time tip.
- `date_parser` - pretvaramo selektirane `String` vrijednosti u prvoj liniji u `datetime` varijable

Detaljnijim pregledom podataka možemo uočiti još jednu stvar koju moramo riješiti.

YYYYMM	MSN	Value	Column_Order	Description	Unit
1973-01-01	CLEIEUS	72.076	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-02-01	CLEIEUS	64.442	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-03-01	CLEIEUS	64.084	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-04-01	CLEIEUS	60.842	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-05-01	CLEIEUS	61.798	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-06-01	CLEIEUS	66.538	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-07-01	CLEIEUS	72.626	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-08-01	CLEIEUS	75.181	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-09-01	CLEIEUS	68.397	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-10-01	CLEIEUS	67.668	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-11-01	CLEIEUS	67.021	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-12-01	CLEIEUS	71.118	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
NaT	CLEIEUS	811.791	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1974-01-01	CLEIEUS	70.55	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1974-02-01	CLEIEUS	62.929	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide

Slika 4. Čišćenje podataka. Izvor: vlastiti rad

Nakon svakog kraja godine, skup podataka sadrži redak sa sumom za tu godinu. Najbolji pristup bi bio da izbacimo indeksne ćelije koje nisu *date time* tipa.

```
ts = data[pd.Series(pd.to_datetime(data.index, errors = 'coerce')).notnull().values]

ts.head(15)
```

YYYYMM	MSN	Value	Column_Order	Description	Unit
1973-01-01	CLEIEUS	72.076	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-02-01	CLEIEUS	64.442	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-03-01	CLEIEUS	64.084	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-04-01	CLEIEUS	60.842	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-05-01	CLEIEUS	61.798	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-06-01	CLEIEUS	66.538	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-07-01	CLEIEUS	72.626	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-08-01	CLEIEUS	75.181	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-09-01	CLEIEUS	68.397	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-10-01	CLEIEUS	67.668	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-11-01	CLEIEUS	67.021	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1973-12-01	CLEIEUS	71.118	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1974-01-01	CLEIEUS	70.55	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1974-02-01	CLEIEUS	62.929	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide
1974-03-01	CLEIEUS	64.519	1	Coal Electric Power Sector CO2 Emissions	Million Metric Tons of Carbon Dioxide

Slika 5. Izvor: vlastiti rad

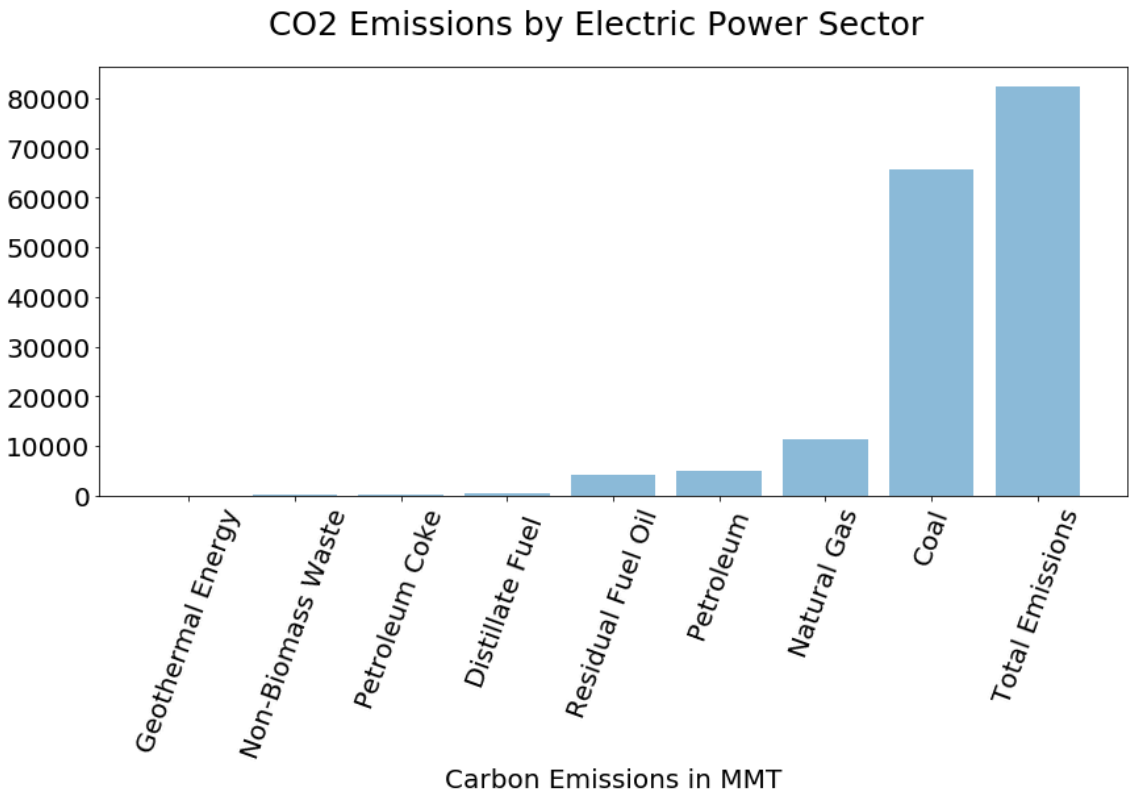
Još ćemo samo promijeniti tip podataka pod kolonom `Value`, naime inicijalno je ona Objekt, no mi želimo da vrijednost koju ćemo promatrati bude numeričkog tipa.

```
ts['Value'] = pd.to_numeric(ts['Value'] , errors='coerce')

ts.dropna(inplace = True) # Izbaci retke bez vrijednosti
```

Vizualizacija podatka

Pod kolonom opis nalazi se podatak o izvoru emisije Ugljikovog dioksida. Kako bi imali bolji uvid u izvore i koliko ih ima, vizualizirat ćemo ih.



Slika 6. Vizualizacija emisija po vrsti. Izvor: vlastiti rad

Možemo vidjeti da su Ugljen i Prirodni plinovi najviše prisutni, pogotovo kada se uzme u obzir koliko ostali pridonose stupcu Ukupna emisija CO2.

Uvijek je preporučljivo raditi ovakve i slične vizualizacije kako bi bolje upoznali naš skup podataka. Vizualizacijom dobivamo okvirnu ideju o prirodi skupa podataka i izuzetno je važna u početnim koracima modeliranja.

Kada se promatraju i predviđaju vremenski nizovi, uglavnom promatramo jednu jedinu varijablu koja se mijenja kroz vrijeme. Tako da ćemo prije nego krenemo u sam postupak modeliranja dataset izmijeniti te ostaviti samo dvije kolone.

Može se pretpostaviti da su te dvije indeksna kolona koja je tipa `datetime` te sama varijabla koju promatramo u stupcu 'Value'. Nakon što smo uspješno napravili čišćenje, vizualizaciju i pripremu podataka slijede metode koje doprinose samoj izradi modela.

3. Model eksponencijalnog izglađivanja (eng. Exponential smoothing)

Prediktivni modeli koristeći metodu exponential smoothing se baziraju na računanju težinskog prosjeka promatranog skupa podataka kroz vrijeme — vremenskog niza. Gdje se tzv. *weight* smanjuje kako promatrana varijabla u vremenskom nizu postaje starija. [2]

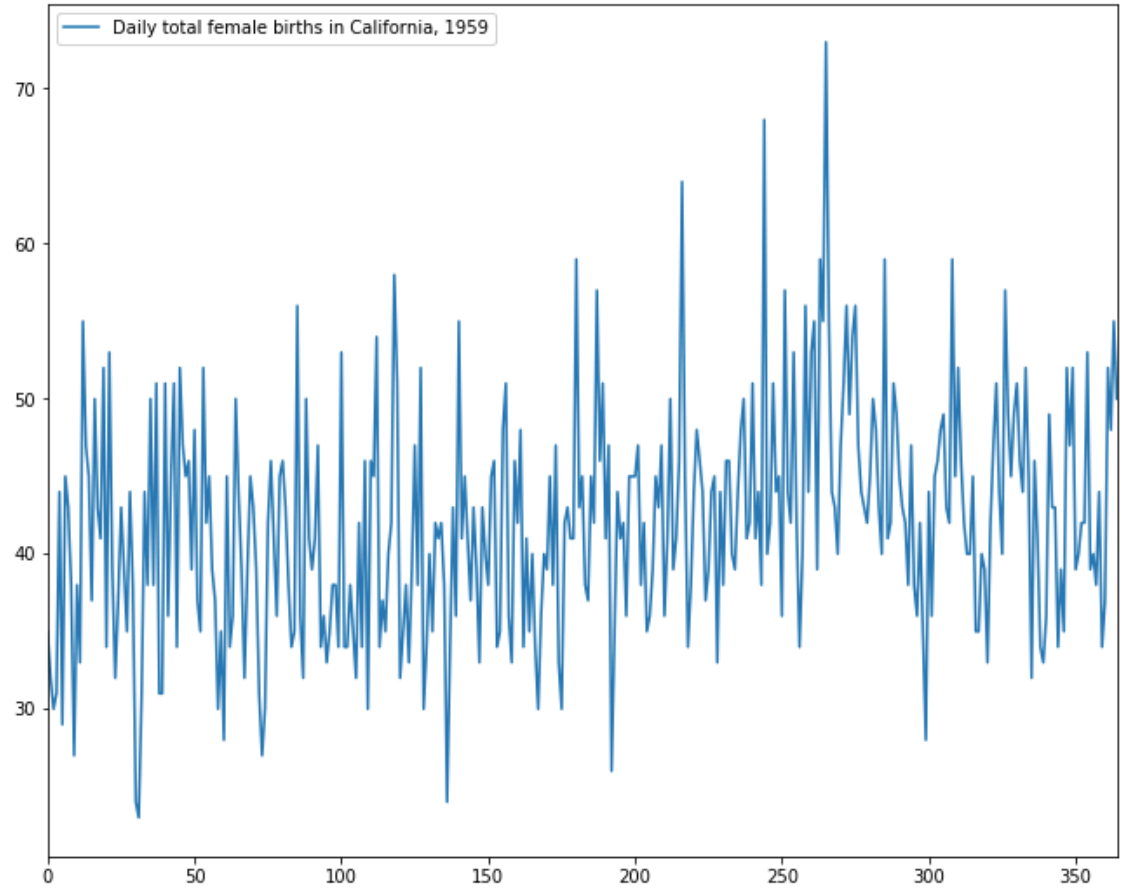
Generalna pretpostavka je da su novije opservacije relevantnije za predviđanje. Tako one sadrže veći *weight*.

Kroz nekoliko metoda proći ćemo kroz najprije jednostavnije postupke izgradnje prediktivnog modela te će se nakon toga graditi i unaprjeđivati model ka složenijim tehnikama. Promatrat ćemo glavne razlike u funkcijama i primjenama svakog modela te dobiti intuiciju za svaki model i generalnu zajedničku ideju što svaki od njih nudi pri prognoziranju.

Pod analizu vremenskih nizova spadaju promatrane varijable uzimajući u obzir vremensku komponentu. Tako želimo uhvatiti i sve moguće relevantne promjene koje utječu na promatranu varijablu u aspektu vremena, sezonalnosti trenda itd.

3.1 Model jednostavnog prosjeka (eng. Simple Average)

Kao početni korak, najprije ćemo vizualizirati skup podataka koji ćemo koristiti kao primjer za izgradnju modela.



Slika 7. Vizualizacija skupa podatka. Izvor: vlastiti rad

Prvo pravilo koje se veže uz model jednostavnog prosjeka jest da se ne primjenjuje u podacima koji nemaju jasan trend. Ovakva metoda ima za pretpostavku da nema jasnog pozitivnog ili negativnog trenda.

Pristup je sljedeći.

Pri izradi ovakvog modela, uzima se za vjerovanje da je varijabla koju predviđamo konstanta. Vrijednosti su centrirane oko prosjeka. Stoga tražimo broj koji adekvatno predstavlja sve promatrane brojeve pod pretpostavkom da nema pozitivnog ili negativnog trenda.

Formalno to možemo zapisati kao:

$$F = a$$

Gdje je a konstanta koju želimo pronaći. Tu se postavlja pitanje, ako predstavlja konstantu, zašto imamo toliko varijacija unutar skupa? Formulu možemo bolje objasniti na sljedeći način:

$$F = a + \varepsilon$$

Gdje je ε šum unutar distribucije koji sadrži parametre. Prvi parametar predstavlja srednju vrijednost koja iznosi 0, dok je drugi parametar varijanca od sigma na kvadrat.

Formalno ε možemo definirati kao:

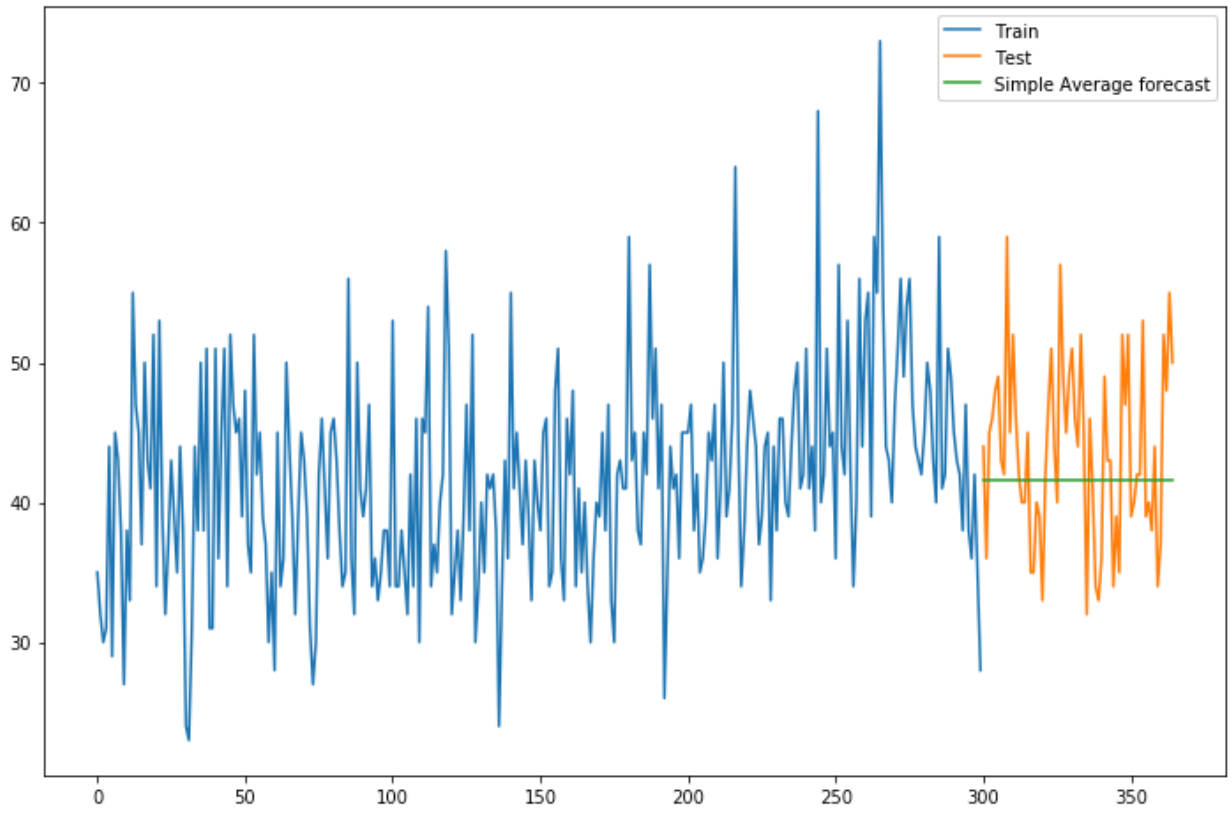
$$\varepsilon = (0, \sigma^2)$$

Model jednostavnog prosjeka je najjednostavniji način na koji možemo predstaviti i predvidjeti vrijednost koristeći ovakav pristup. Unutar ove metode:

- ne izostavljamo nikakve podatke, sve ulazne opservacije se uzimaju u obzir pri izradi modela
- pridružujemo jednak *weight* svim točkama unutar distribucije, drugim riječima svaka varijabla kroz vremenski niz jednako je važna za procjenu sljedeće točke u budućnost

Dakle, ova metoda zaključuje da sve promatrane vrijednosti imaju jednaku važnost. Vrlo često to nije poželjan slučaj promatranja stvari. Pogotovo kada imamo veliku povijest podataka i uzmemo u obzir da se okruženje s vremenom mijenja te se model tome treba prilagoditi.

```
train=data[0:300]
test=data[300:]
y_hat_avg = test.copy()
y_hat_avg['SA_forecast'] = train['Daily total female births in California,
1959'].mean()
plt.figure(figsize=(12,8))
plt.plot(train, label = 'Train')
plt.plot(test, label = 'Test')
plt.plot(y_hat_avg['SA_forecast'], label = 'Simple Average forecast')
plt.legend(loc='best')
plt.show()
```



Slika 8. Vizualizacija metode jednostavnog prosjeka. Izvor: vlastiti rad

Valja napomenuti da je u nekim slučajevima ova metoda ponekad i najbolja, unatoč svojoj jednostavnosti. Možemo vidjeti kako je prediktivna linija konstantna, dobili smo središnju vrijednost skupa.

3.2 Model pomičnog prosjeka. (eng. Moving Average)

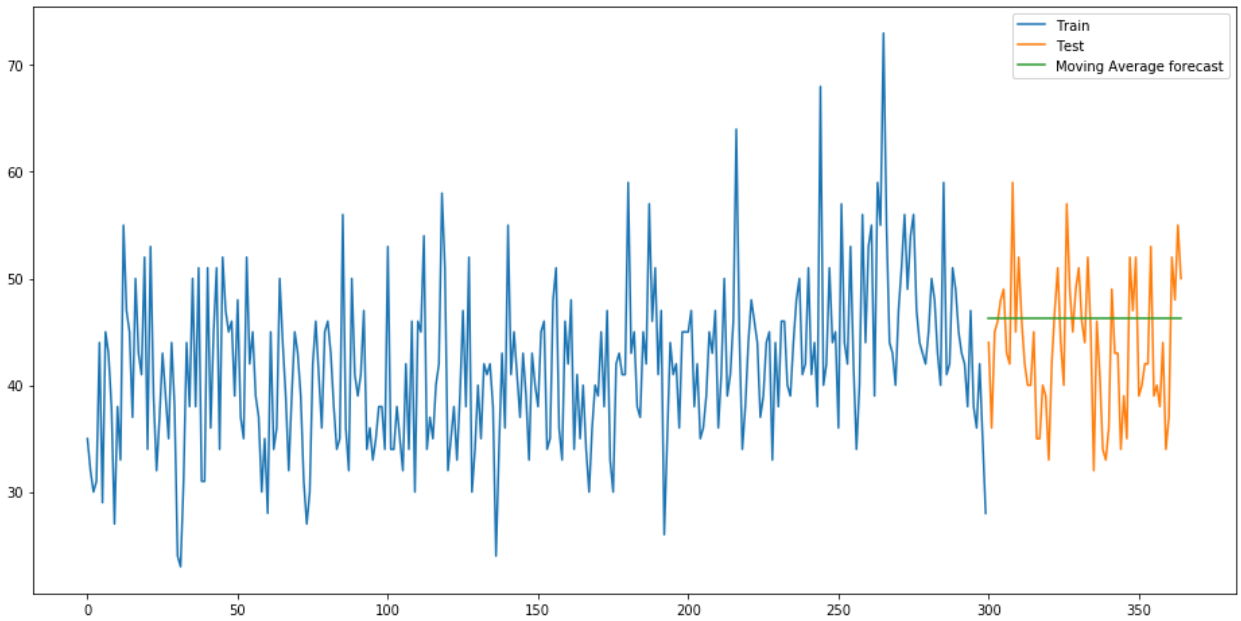
Model pomičnog prosjeka se u jednoj stvari odmiče od prethodnog modela. Ideja je da neke podatke koji su zastarjeli odbacimo u promatranju i izračunamo srednju vrijednost za k vrijednosti.

Primjerice, ako promatramo graf na početku vidimo da su vrijednosti niže nego kasnije u distribuciji. Tako možemo uzeti prosjek određenih perioda za koje mislimo da su relevantni za procjenu budućeg razdoblja.

Koristeći model pomičnog prosjeka, predviđamo na temelju prosjeka konačnog broja prošlih k vrijednosti.

Intuitivno je da je od najveće važnosti pri izradi ovog modela odabir optimuma k.

```
y_hat_avg = test.copy()
y_hat_avg['moving_avg_forecast']  = train['Daily total female births in
California, 1959'].rolling(100).mean().iloc[-1]
plt.figure(figsize=(16,8))
plt.plot(train, label = 'Train')
plt.plot(test, label = 'Test')
plt.plot(y_hat_avg['moving_avg_forecast'], label = 'Moving Average forecast')
plt.legend(loc='best')
plt.show()
```



Slika 9. Model pomičnog prosjeka. Izvor: vlastiti rad

```
rms = sqrt(mean_squared_error(test.Total, y_hat_avg.moving_avg_forecast))
print(rms)
```

```
6.427508190462424
```

U ovom primjeru, uzeli smo 100 zadnjih vrijednosti test skupa podataka kao mjerodavne pri izradi modela. Vidimo da je vrijednost viša od Simple Average.

3.3 Metoda jednostavnog eksponencijalnog izglađivanja (eng. Simple exponential smoothing)

Ono što ova metoda pokušava donijeti jest alternativu i sredinu između Naive metode i Jednostavnog prosjeka. Naive model gleda zadnju vrijednost kao jedinu relevantnu, dok SA gleda sve jednako relevantne.

To pitanje nije trivijalno, treba li neke podatke skroz izbaciti pri izradi modela? Ako da, gdje povući crtu? Kako raspodijeliti *weight* unutar distribucije?

U ovom poglavlju proći ćemo kroz tu problematiku i na kraju vidjeti rezultat te kako se on uspoređuje s prijašnjom metodom.

Postavlja se pitanje da li treba ignorirati neke vrijednosti ili one ipak imaju neku određenu važnost u estimaciji modela.

Metoda jednostavnog eksponencijalnog izglađivanja leži na principu da noviji podaci imaju veću vrijednost i težinu nego promatrane varijable u daljoj prošlosti. Gdje se *weight * smanjuje kako podaci dolaze dalje u prošlost. Najstarija opservacija tako ima najmanju vrijednost. [2]

Formula za ovakvu metodu jest sljedeća:

$$F_t +_1 = aD_t + (1 - a)F_t$$

Gdje su redom:

F_t+_1 — Buduća vrijednost koju pokušavamo izračunati

a — Smoothing constant - vrijednost koja može poprimiti vrijednost $-1 < a < 1$

D_t — Vrijednost u periodu t, koju promatramo trenutno u skupu podataka

F_t — prognozirana vrijednost za period t

Konkretno, D_1 je najstarija vrijednost u nizu. Ako primjerice u skupu imamo 200 promatranih varijabli, onda možemo reći da računamo vrijednost za F201

Stoga možemo postupak prikazati na sljedeći način:

$$F201 = aD200 + (1 - a)F200$$

$$F200 = aD199 + (1 - a)F199$$

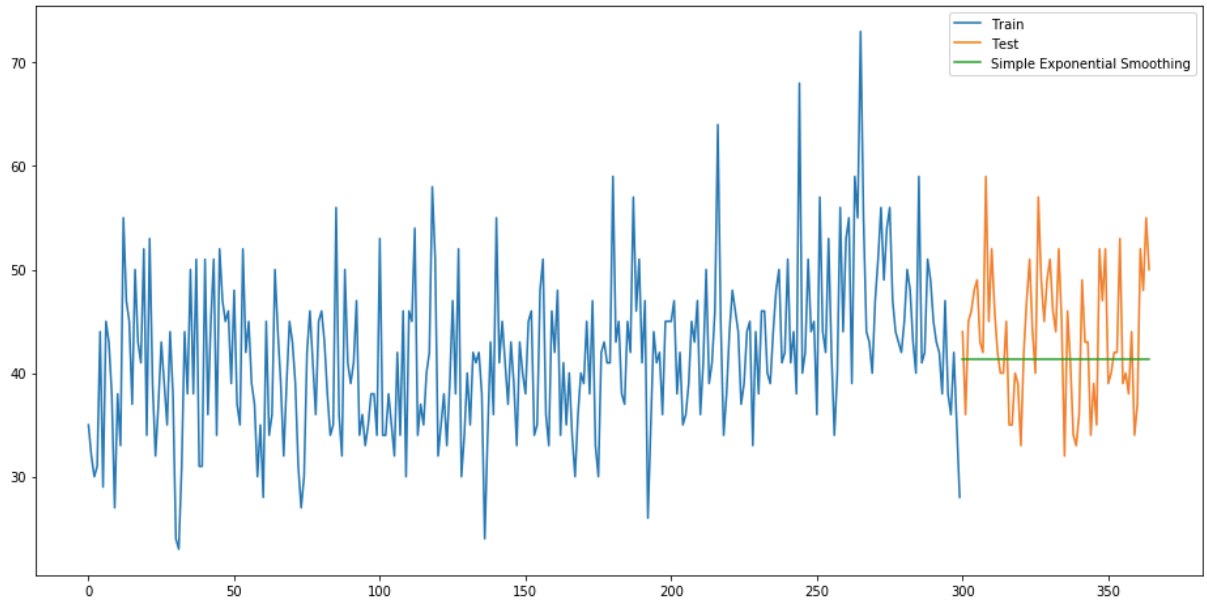
$$\dots$$

S ovakvim pristupom možemo reći da koristeći model jednostavnog eksponencijalnog izglađivanja želimo:

1. Koristiti sve vrijednosti za kreiranje modela
2. Postepeno povećavati važnost vrijednostima koje su novije kroz vremenski period

```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
y_hat_avg = test.copy()
fit_ses =
```

```
SimpleExpSmoothing(np.asarray(train['Total'])).fit(smoothing_level=0.1,
optimized = False)
y_hat_avg['SES'] = fit_ses.forecast(len(test))
plt.figure(figsize=(16,8))
plt.plot(train['Total'], label = 'Train')
plt.plot(test['Total'], label = 'Test')
plt.plot(y_hat_avg['SES'], label = 'Simple Exponential Smoothing')
plt.legend(loc='best')
plt.show()
rms = sqrt(mean_squared_error(test.Total, y_hat_avg.SES))
print(rms)
```



Slika 10. Model jednostavnog eksponencijalnog izglađivanja. Izvor: vlastiti rad

```
rms = sqrt(mean_squared_error(test.Total, y_hat_avg.SES))
print(rms)
```

```
6.6834360976039155
```

Kao što vidimo, model pomičnog prosjeka koja je puno jednostavnija metoda dala je točnije rezultate za određeni postotak. Ovakva metoda nekada može biti najbolja za odabrati, pogotovo kada noviji podaci imaju veliki utjecaj na kratkoročno buduće kretanje niza.

3.4 Holtsova Linearna Trend metoda

Ono što metoda jednostavnog eksp. izglađivanja radi u podacima koji nemaju izražen trend, Holts Linear Trend metoda radi kada je trend prisutan u skupu. Dakle nadograđujemo prethodnu tehniku te pritom uzimamo u obzir postojanje trenda.

Holts zadržava pristup oko dodjeljivanja weighta na isti način, međutim ima dodatne komponente, posebice snop koji je vezan i za linearnu regresiju. [2] Želi se iterativno kroz proces predvidjeti i izračunati eventualan trend koji se pojavljuje, bio pozitivan ili negativan.

Imamo formulu:

$$F_t + 1 = a_t + b_t$$

Gdje se vrijednosti a_t i b_t periodički definiraju sljedećim formulama:

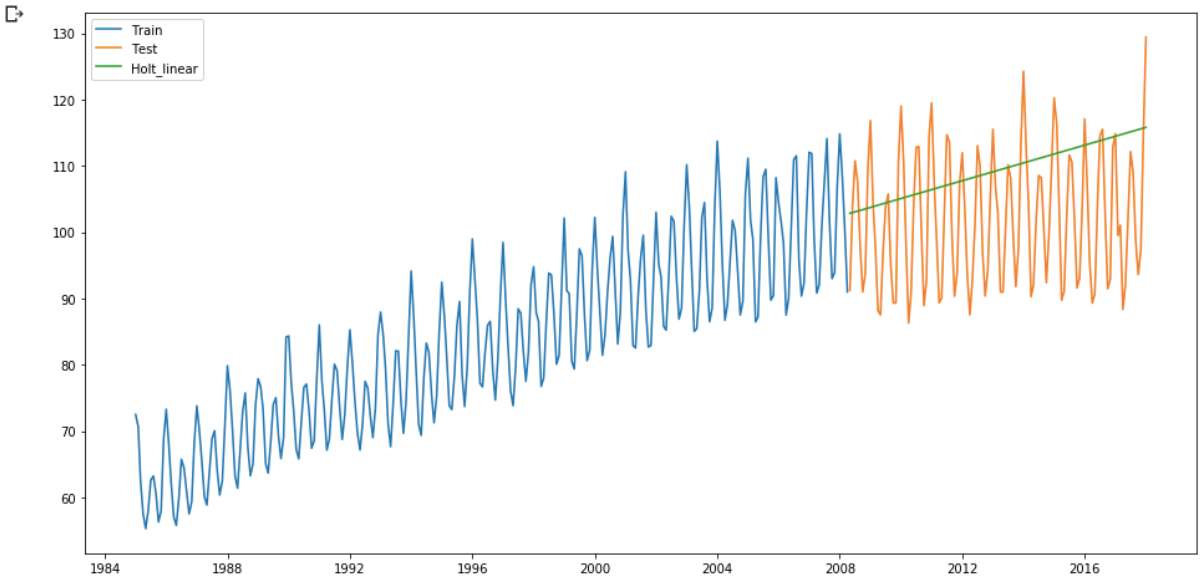
$$a_t = aD_t + (1-a)(a_{t-1} + b_{t-1})$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

Formula sadrži dvije jednadžbe za parametar izglađivanja.

Prva se odnosi na level, dok je druga za trend unutar vremenskog niza. Formula za level sadržana je prethodnoj metodi.

```
y_hat_avg = test.copy()
fit_holt =
Holt(np.asarray(train['IPG2211A2N']).astype('float64')).fit(smoothing_level =
0.1, smoothing_slope = 0.1)
y_hat_avg['Holt_linear'] = fit_holt.forecast(len(test))
plt.figure(figsize=(16,8))
plt.plot(train['IPG2211A2N'], label='Train')
plt.plot(test['IPG2211A2N'], label='Test')
plt.plot(y_hat_avg['Holt_linear'], label='Holt_linear')
plt.legend(loc='best')
plt.show()
```



Slika 11. Holtsova linearna trend metoda. Izvor: vlastiti rad

Promijenili smo skup podataka jer nije postojao trend u prijašnjem, te sljedeće dvije metode nisu prigodne za takav skup. Ovdje možemo vidjeti jasan pozitivan trend kroz vrijeme uz jaku sezonalnost. Točno se može primijetiti kako sada imamo i trend, dakle prediktivne vrijednosti više nisu konstantne.

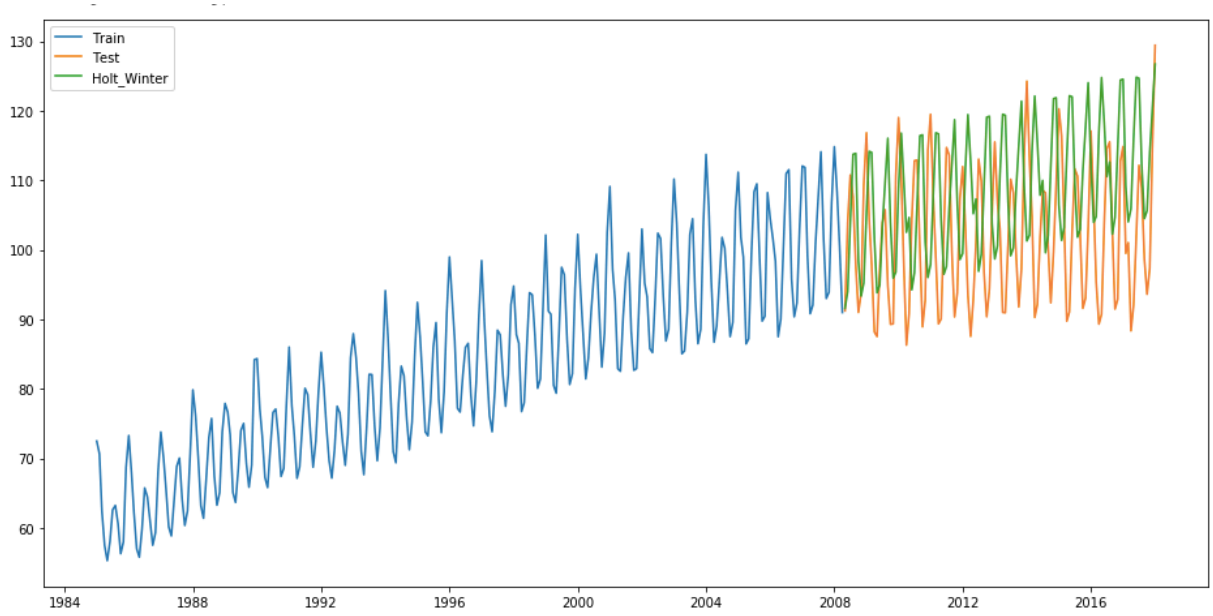
3.5 Holt-Winters model

Radi se o nadograđenoj verziji prijašnjeg modela. Rekli smo da se u Holts linearnom modelu dodala komponenta vezana za trend koji uzimamo u obzir. U Holt-Winters metodi želimo uz trend uzeti u obzir i sezonalnost skupa podataka.

S tim da imamo dva pristupa, aditivni i multiplikativni, ovisno kako se ponaša sezonalnost kroz skup.

Na gore dvije navedene formule dodaje se i γ kao parametar izglađivanja te s_t - sezonalnost skupa.

```
y_hat_avg = test.copy()
fit_hw =
ExponentialSmoothing(np.asarray(train['IPG2211A2N']).astype('float64'),
seasonal_periods =25, trend = 'add', seasonal = 'add').fit()
y_hat_avg['Holt_Winter'] = fit_hw.forecast(len(test))
plt.figure(figsize=(16,8))
plt.plot( train['IPG2211A2N'], label='Train')
plt.plot(test['IPG2211A2N'], label='Test')
plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
plt.legend(loc='best')
plt.show()
```



Slika 12. Holt-Winters model. Izvor: vlastiti rad

Možemo vidjeti kako ovakva metoda daje najbolje rezultate. U jednadžbu smo uzeli sve komponente vremenskog niza, te je tako prediktivni model točniji.

4. ARIMA model

ARIMA (autoregressive integrated moving average) je skup metoda koji su do neke razine generalizirani. S ARIMA modelom uzimamo autokorelaciju unutar niza i pokušavamo ju direktno modelirati. [2]

Do sada smo prošli kroz nekoliko različitih metoda i pristupa koji se mogu koristiti za predviđanje vremenskih nizova na temelju njegove povijesti. To su metode poput regresije i eksponencijalnog izglađivanja.

Tako možemo promatrati sve te različite metode kao skup različitih, specifičnih i uskih metoda koje možemo koristiti zajedno ili zasebno u svrhu izrade što boljeg modela. Međutim, postoji metoda koja objedinjuje sve te manje tipove modela u jedan više generalan pristup izradi modela kod vremenskih nizova. Takav model poznat je kao ARIMA model. [2]

ARIMA sadrži sistematičan niz pravila i pristupa s kojim kada primijenimo razne modele možemo odlučiti koji je najpogodniji za taj konkretan slučaj.

Najprije ćemo vizualizirati ukupan skup kako bi dobili sliku o prirodi vremenskog niza i širu sliku kako se skup ponaša kroz vrijeme. Promatranjem skupa zabilježit ćemo podatke o trendu, sezonalnosti.

Prije izrade samog modela potrebno je provesti nekoliko koraka kako bi se uvjerili da su podaci i niz onakvi kakvi želimo.

Otprilike koraci izgledaju ovako:

5 koraka za ARIMA model:
1. Vizualizacija vremenskog niza
2. Stacioniranje niza
3. Izrada ACF/PACF grafa i pronalazak optimalnog parametra
4. Izrada ARIMA modela
5. Izrada predikcija (budućih vrijednosti) na temelju izgrađenog i validiranog modela

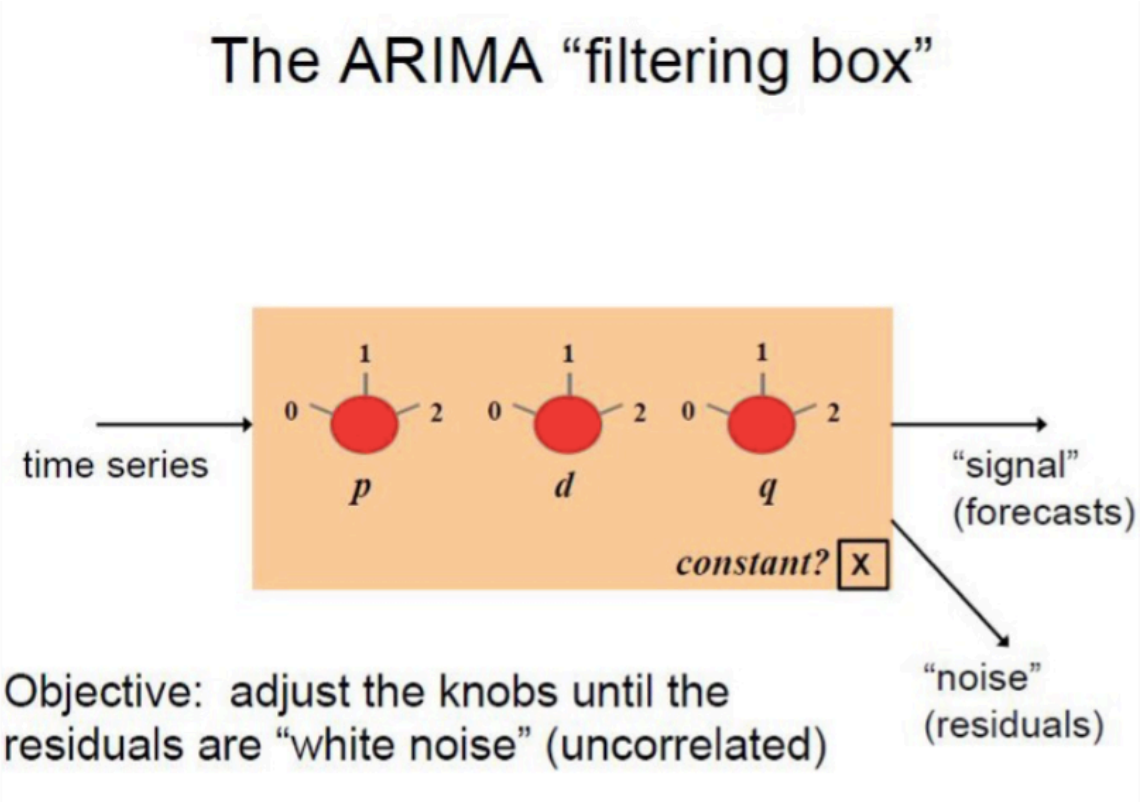
U ovom poglavlju opisat ćemo i napraviti svaki od navedenih koraka, te će krajnji rezultat tih metoda biti ARIMA model.

ARIMA model može se opisati s 3 parametra:

p - broj autoregresivnih uvjeta (odnosi se na AR u ARIMA modelu). Dopušta nam da uklopimo na ispravan način efekte prošlih povijesnih vrijednosti u naš model.

d - broj nesezonalnih razlika. Odnosni se na stacionarnost, pokušava uhvatiti uzorke u sezonalnošću i te uzorke uzeti u obzir pri izgradnji modela u budućnost.

q - broj pomičnih prosjeka. Pomoću q parametra postavljamo tzv. pogrešku našeg modela u odnosu na prošle vrijednosti. (MA dio)



Slika 13. Preuzeto sa: Unboxing ARIMA Models, Madhav Mishra [https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8]

4.1 Stacionarnost i nestacionarnost vremenskih nizova

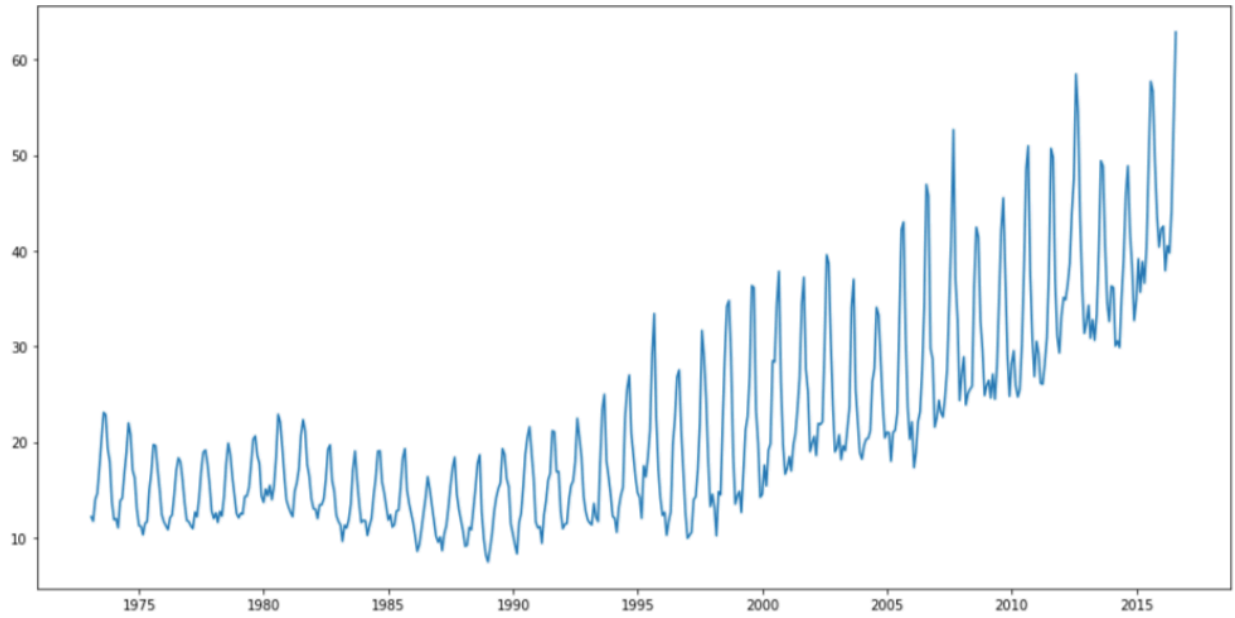
Stacionarnost je jako bitna komponenta vremenskih nizova te je treba uzeti u obzir pri svakoj tehnici modeliranja. Neki modeli zahtijevaju pretvorbu stacionarnih u nestacionarne vremenske nizove.

Postoje tri osnovna kriterija u vezi vremenskih nizova koji mogu takav niz svrstati u stacionarni.

1. Srednja vrijednost bi trebala biti konstanta, ne funkcija.
2. Varijanca se ne bi trebala kroz vrijeme mijenjati već držati oko jednog raspona vrijednosti koji je konstantan.
3. Kovarijanca bi trebala biti konstantna kroz vrijeme.

Sve dok je naš vremenski niz nestacionaran, ne možemo razvijati prediktivni model. U slučajevima gdje je ona prisutna, potrebno je prvotno stacionirati vremenski niz te zatim primjenjivati stohastičke modele za predikciju. Postoji više načina koji mogu model pretvoriti u stacionaran. To su metode poput *Detrendinga*, *Differencinga* itd.

Prvo moramo vizualizirati naš vremenski niz odnosno podatke. Iz grafa ćemo dobiti okvirnu sliku trenda, sezonalnosti i kako se promatrana varijabla ponaša kroz vrijeme. Nakon toga možemo prijeći na spomenute metode koje se bave trendom i sezonalnošću.



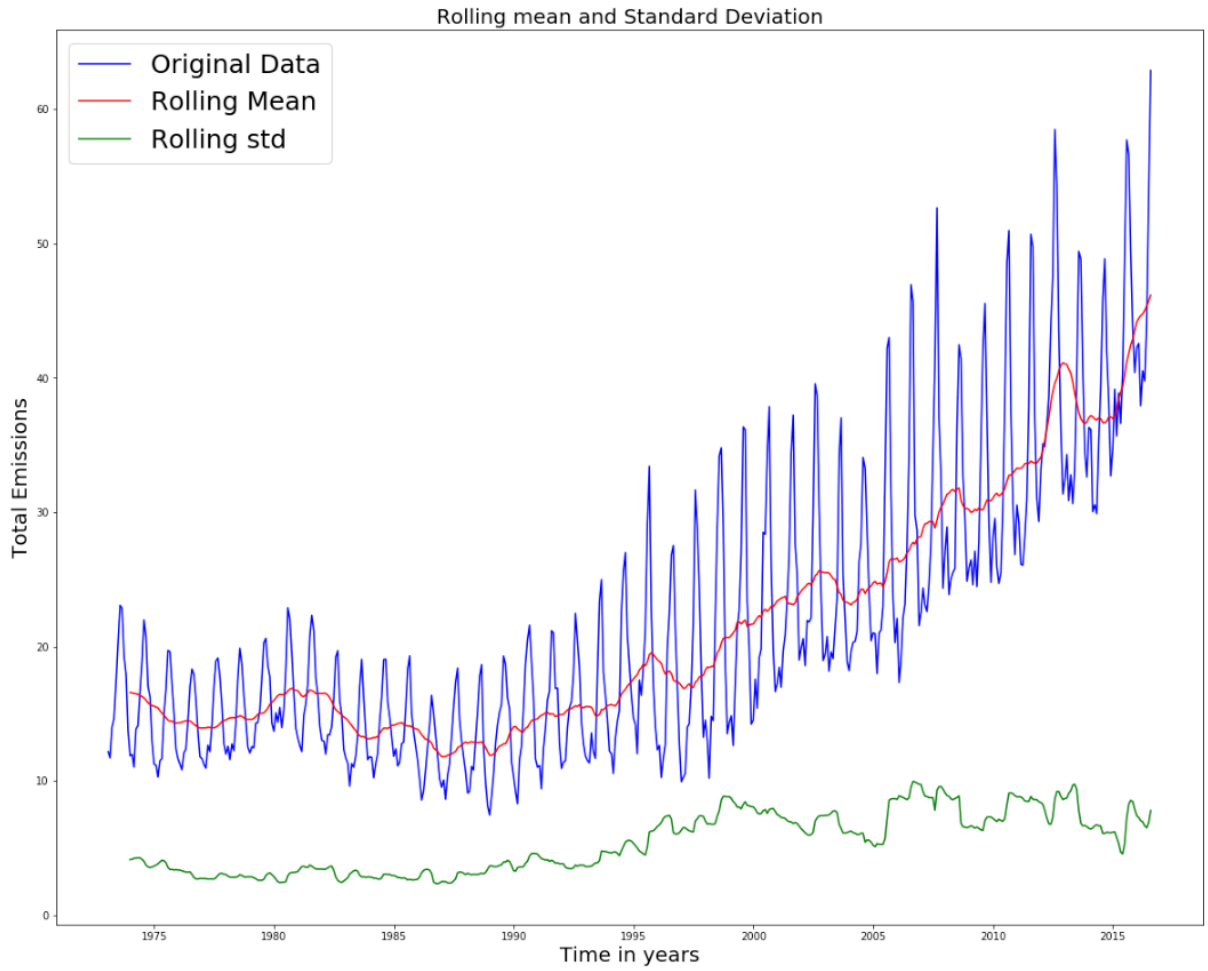
Slika 14. Vizualizaicja skupa. Izvor: vlastiti rad

Evidentno je da postoji trend u CO2 emisiji kroz vrijeme kao i sezonalnost koja se ponavlja. Možemo zaključiti da promatrani vremenski niz nije stacionaran.

Dickey-Fuller test

Postoji i formalan način testiranja stacionarnog skupa podataka. Polazna hipoteza jest da je dani skup nestacionaran. Rezultati se uspoređuju s kritičnom vrijednosti u različitim intervalima pouzdanosti. Ako su rezultati testa manji od kritične vrijednosti odbacujemo nultu hipotezu i tada možemo reći da je vremenski niz stacionaran.

Vodeći se gore navedenim kriterijima koji se vežu za stacionarnost i nestacionarnost, vizualizirat ćemo središnju vrijednost i varijancu kroz cijeli vremenski niz i vidjeti kako se ponaša i kreće kroz samu funkciju



Slika 15. Ispitivanje stacionarnosti Izvor: vlastiti rad

Jasno se vidi na grafu kako i srednja vrijednost i standardna devijacija variraju kroz vrijeme. Već znamo da niz ima trend i sezonalnost.

Uz vizualizaciju možemo to prikazati i standardnim testom.

```
def TestStationaryAdfuller(ts, cutoff = 0.01):
    ts_test = adfuller(ts, autolag = 'AIC')
    ts_test_output = pd.Series(ts_test[0:4], index = ['Test Statistic', 'p-value',
'#Lags Used', 'Number of Observations Used'])

    for key, value in ts_test[4].items():
        ts_test_output['Critical Value (%s)'%key] = value
    print(ts_test_output)

    if ts_test[1] <= cutoff:
        print("Strong evidence against null hypothesis")
    else:
        print("Weak evidence against null hypotesis, time series is non-stationary")

TestStationaryAdfuller(mte)
```

```
Test Statistic                   1.831215
p-value                          0.998409
#Lags Used                      19.000000
Number of Observations Used    503.000000
Critical Value (1%)             -3.443418
Critical Value (5%)             -2.867303
Critical Value (10%)            -2.569840
dtype: float64
Weak evidence against null hypotesis, time series is non-stationary
```

Transformacija podataka u stacionaran niz

Kako bi nastavili s razvijanjem prediktivnog modela, moramo niz pretvoriti u stacionaran. Najčešće korištene tehnike koje estimiraju trend u modelu i uklanjaju su:

Agregacija - uzimanje prosjeka kroz određeni vremenski period

Izglađivanje - uzimanje kotrljajućeg prosjeka

Polinom - postaviti model regresije

Uzeti ćemo srednju vrijednost od 'k' uzastopnih vrijednosti uzimajući u obzir frekvenciju vremenskog niza (u ovom slučaju 12 mjeseci po godini).

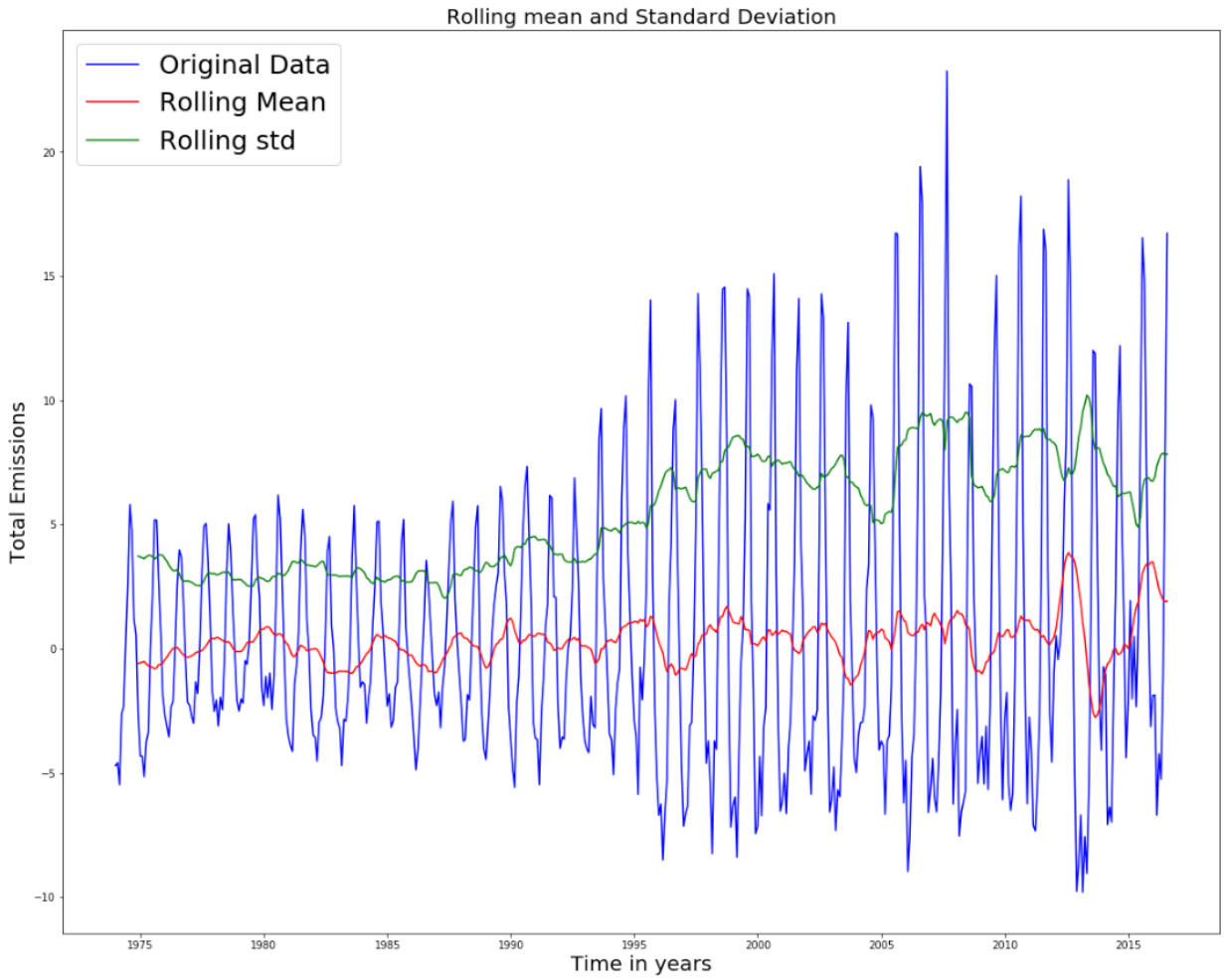
```
mte_MA_diff = mte - moving_avg
mte_MA_diff.head(15)
```

```
YYYYMM
1973-01-31         NaN
1973-02-28         NaN
1973-03-31         NaN
1973-04-30         NaN
1973-05-31         NaN
1973-06-30         NaN
1973-07-31         NaN
1973-08-31         NaN
1973-09-30         NaN
1973-10-31         NaN
1973-11-30         NaN
1973-12-31   -4.705333
1974-01-31   -4.594333
1974-02-28   -5.475083
1974-03-31   -2.624917
Freq: M, Name: Natural Gas Electric Power Sector CO2 Emissions, dtype: float64
```

Vidimo da su izbrisane vrijednosti prvih godinu dana odnosno naš određen 'k'.

Kako bi vidjeli kako sada stvari stoje možemo ponovno testirati stacionarnost.

```
mte_MA_diff.dropna(inplace=True)
TestStationaryPlot(mte_MA_diff)
```



Slika 16. Metoda pomicanja vrijednosti. Izvor: vlastiti rad

Odmah se vidi jasna razlika u odnosu na test napravljen prije. Kada napravimo i drugu varijantu testa dobijemo:

```
Test Statistic                  -5.138977
p-value                          0.000012
#Lags Used                      19.000000
Number of Observations Used    492.000000
Critical Value (1%)             -3.443711
Critical Value (5%)             -2.867432
Critical Value (10%)            -2.569908
dtype: float64
Strong evidence against null hypothesis
```

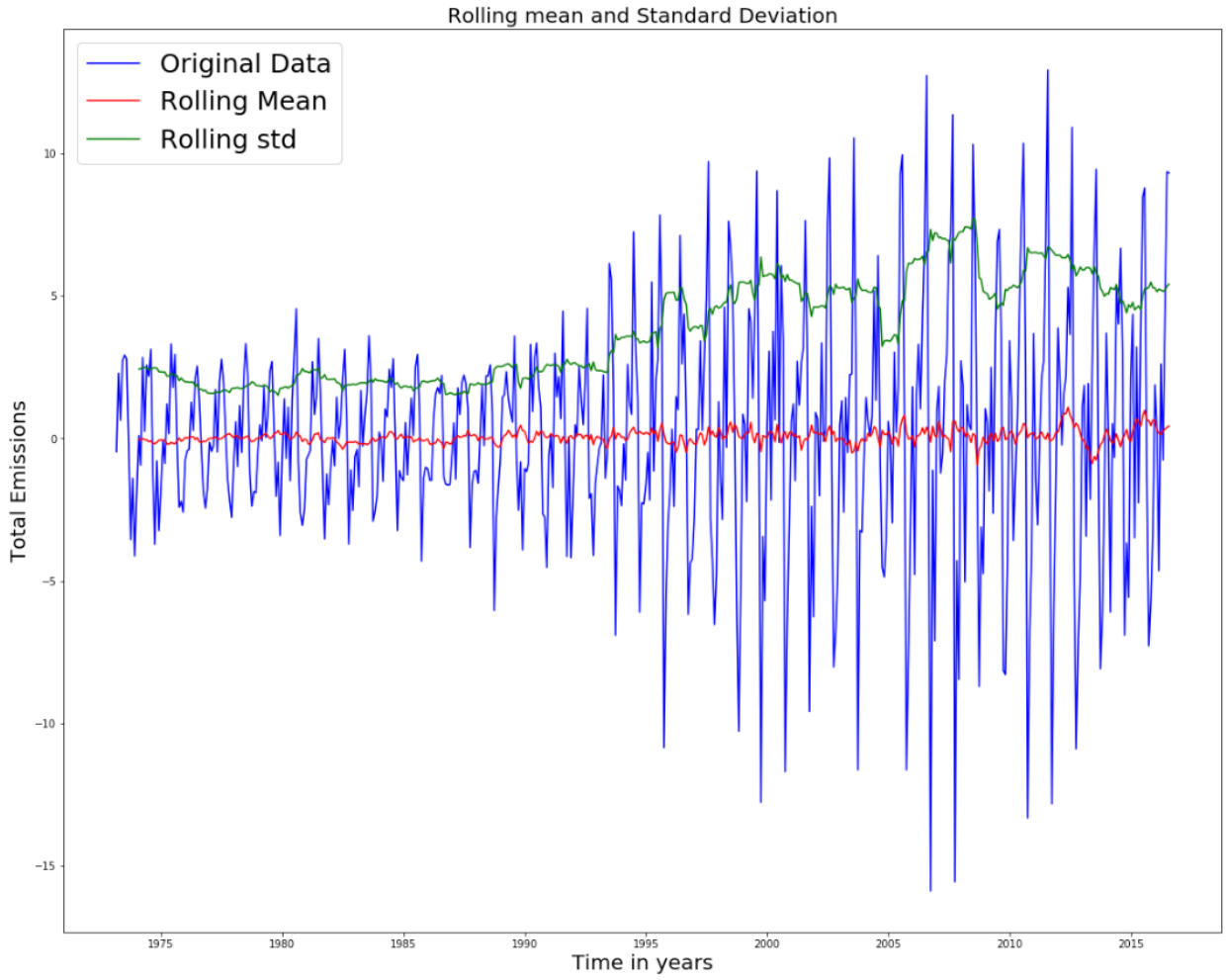
Možemo reći s 99% pouzdanosti da je niz stacionaran.

Diferenciranje

Radi se metodi koja je najčešće upotrebljavana kao alat za uklanjanje trenda i sezonalnosti. U ovoj metodi uzimamo razliku između originalne promatrane varijable u određenom trenutku s tom istom vrijednosti u prethodnom trenutku.

Diferenciranje prvog reda može se napraviti na sljedeći način:

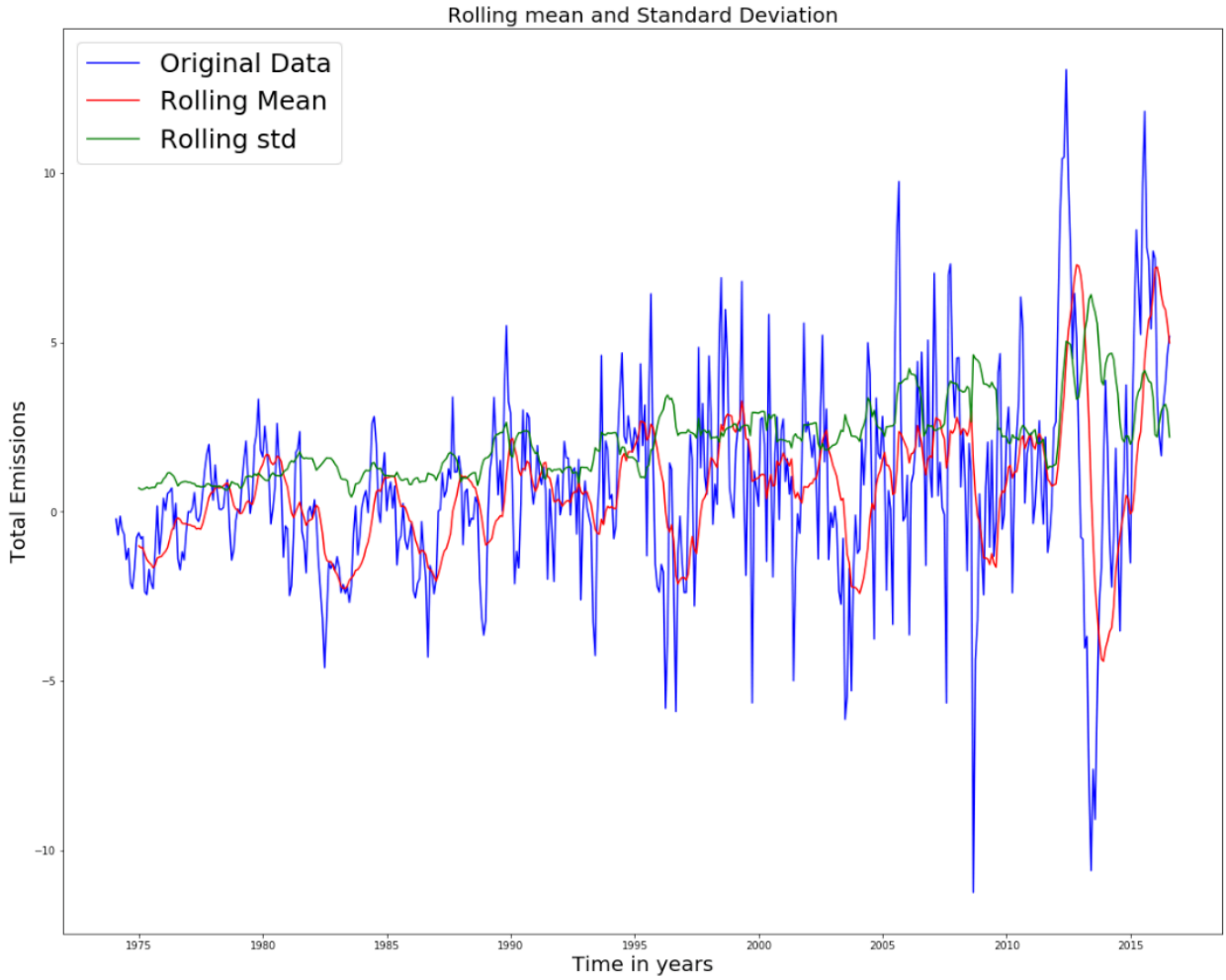
```
mte_first_difference = mte - mte.shift(1)
TestStationaryPlot(mte_first_difference.dropna(inplace=False))
```



Slika 17. Metoda diferenciranja. Izvor: vlastiti rad

Možemo vidjeti kako je značajno popravilo svojstvo stacionarnosti u skupu. Rekli smo da uz trend možemo ukloniti i sezonalnost.

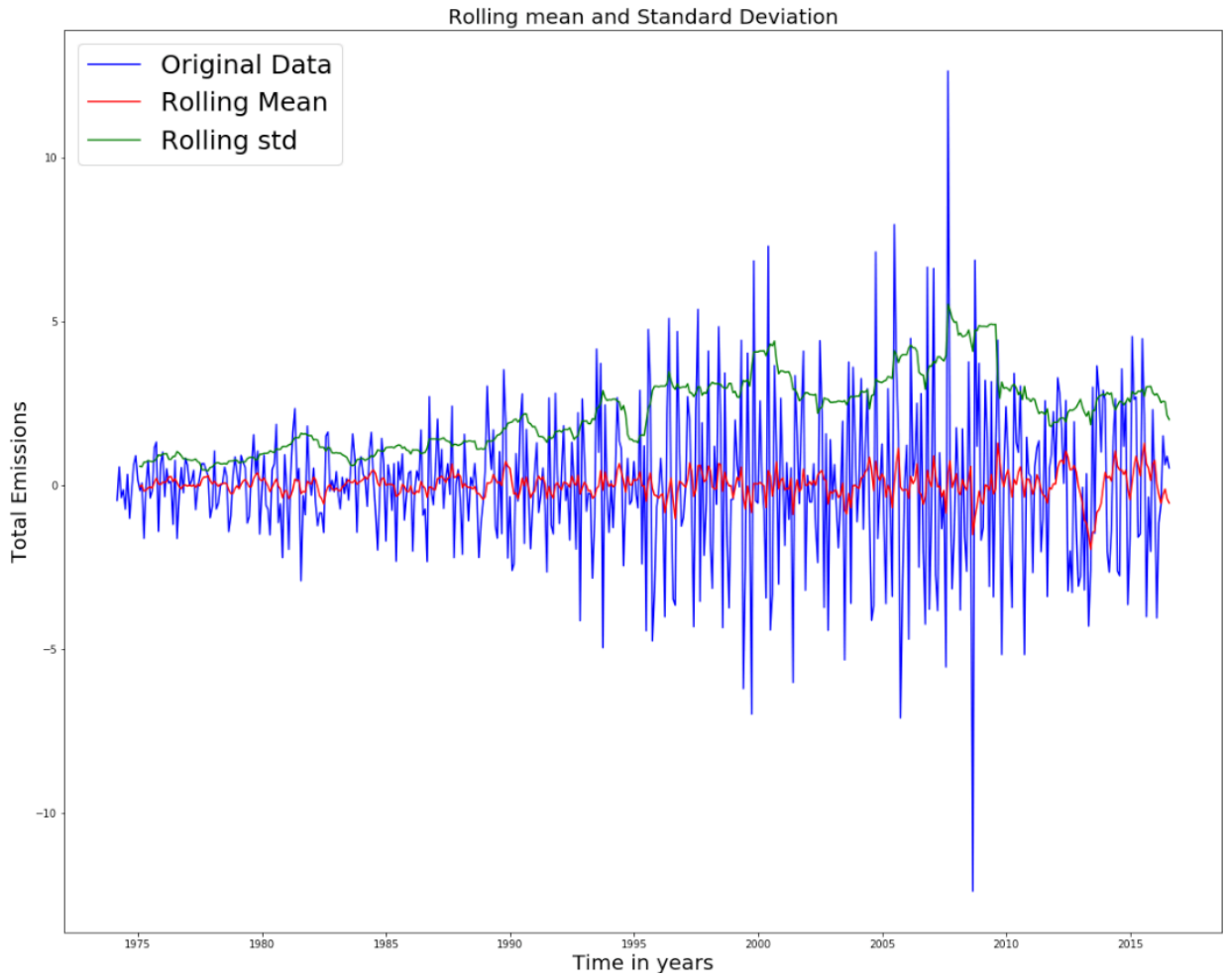
```
mte_seasonal_difference = mte - mte.shift(12)
TestStationaryPlot(mte_seasonal_difference.dropna(inplace=False))
TestStationaryAdfuller(mte_seasonal_difference.dropna(inplace=False))
```



Slika 18. Testiranje stacionarnosti 2. Izvor: vlastiti rad

Vidimo da se uzimajući u obzir i sezonalnost poboljšala stacionarnost niza. Možemo otiči korak dalje i uzeti prvi stupanj diferencije kroz sezonalno diferenciranje.

```
mte_seasonal_first_differece = mte_first_difference - mte_first_difference.shift(12)
TestStationaryPlot(mte_seasonal_first_difference.dropna(inplace=False))
```



Slika 19. Testiranje stacionarnosti 3 Izvor: vlastiti rad

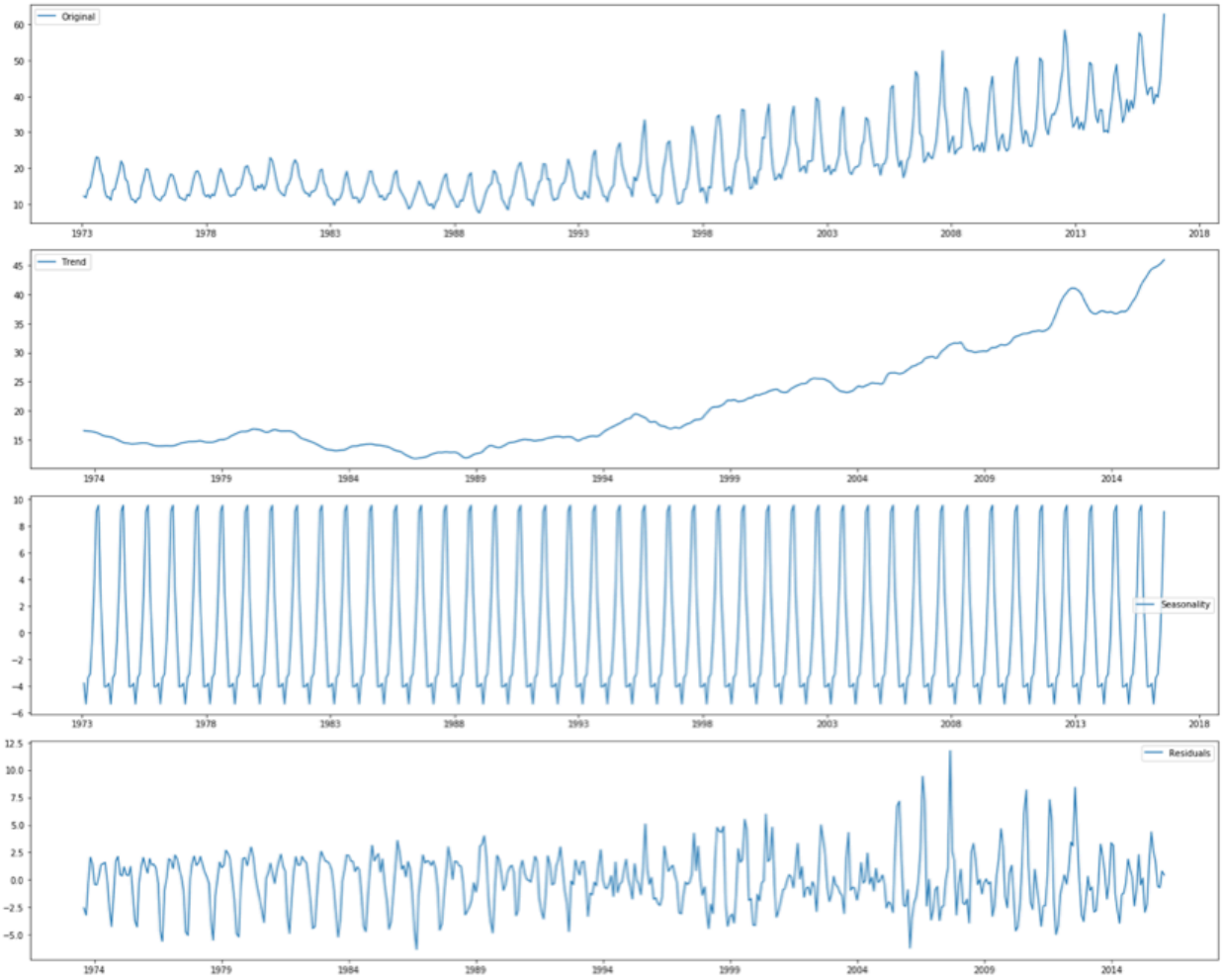
Ovakav pristup dao je najbolju statičnost našem nizu. Rolling mean jako dobro prolazi i opisuje srednju vrijednost kroz vremenski niz.

4.2 Dekompozicija vremenskog niza

Dekompozicija vremenskog niza odnosi se na razdvajanje i raščlanjivanje komponenti koje zajedno tvore distribuciju vremenskog niza. Te komponente su trend koji je sporomijenjajuće te može imati tendenciju rasta ili pada, sezonalnost i šum.

U svrhu boljeg shvaćanja, izgradnje i manipulacije česta je metoda dekompozicije modela.

Kada dekomponiramo model i razdijelimo na trend, sezonalnost i ostatak dobijemo:



Slika 20. Dekompozicija vremenskog niza. Izvor: vlastiti rad

Ono što možemo modelirati jest ostatak odnosno šum, podatke smo razdvojili od trenda i sezonalnosti.

Prije nego krenemo modelirati bilo bi dobro provjeriti stacionarnost niza.

```
mte_decompose = residual
mte_decompose.dropna(inplace = True)
TestStationaryAdfuller(mte_decompose)
```

```
Test Statistic                -8.547084e+00
p-value                        9.439345e-14
#Lags Used                     1.900000e+01
Number of Observations Used    4.910000e+02
Critical Value (1%)           -3.443739e+00
Critical Value (5%)           -2.867444e+00
Critical Value (10%)          -2.569915e+00
dtype: float64
Strong evidence against null hypothesis
```

Kada dekomponiramo dijelove i uklonimo trend i sezonalnost koji se mogu objasniti i interpretirati kroz vremenski niz ostaje nam bijeli šum. Od ostatka niza očekujemo da autokorelativna mjera bude blizu nule, u protivnom možemo zaključiti da nismo skroz razdvojili tren i sezonalnost jer još uvijek ima određenih uzoraka koji se mogu interpretirati.

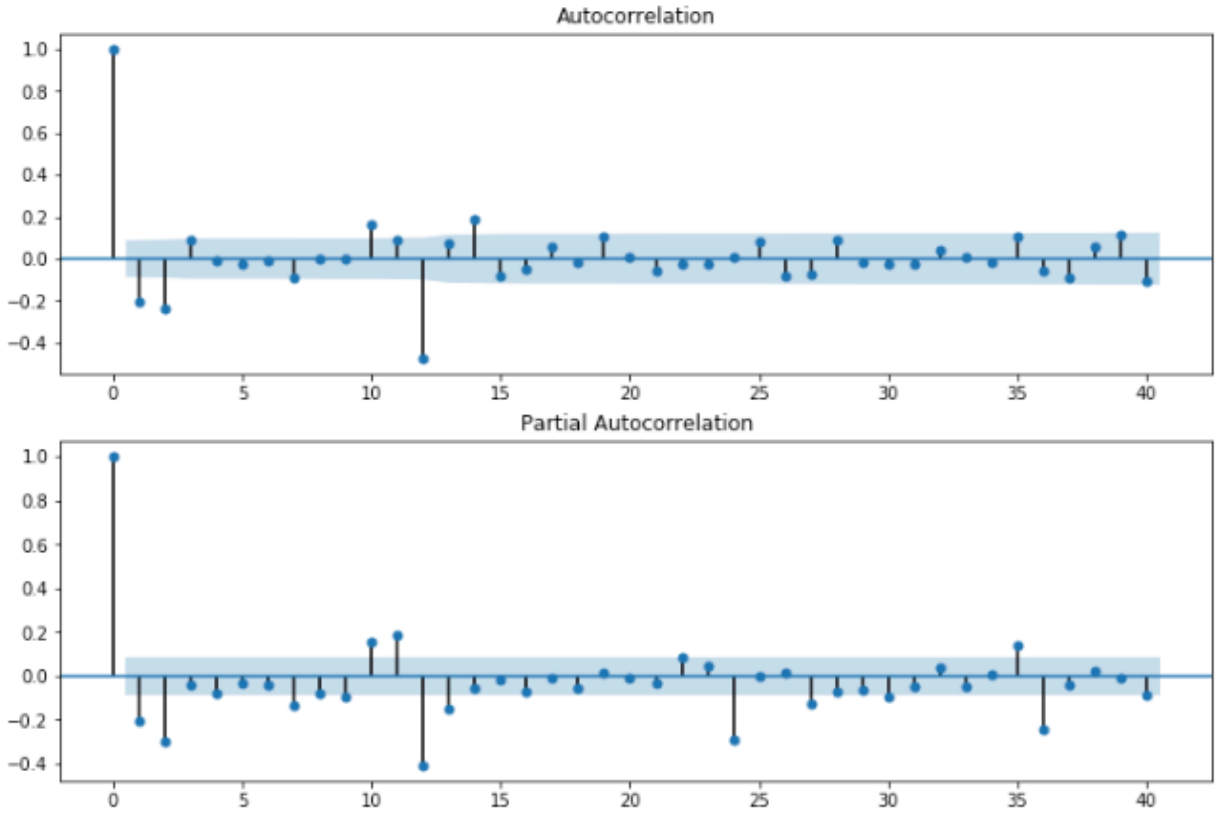
Bijeli šum teži prema konstantnoj srednjoj vrijednosti i obično ima konstantu varijancu.[2]

4.3 Identifikacija modela i pronalaženje optimalnih parametara

Kada razvijamo model za predikciju vremenskog niza cilj nam je naći vrijednosti ARIMA parametara:

Parametri	
p	AR odnosi se na *lagged* vrijednosti koji se odnosi na efekt prošlih vrijednosti na naš model. Primjerice da li x(t) ovisi o x(t-5).
q	Broj od Moving Average uvjeta. Slično kao kod slučaja iznad, uzimamo prošle vrijednosti i pomičemo ih u našem nizu kao prediktivnu funkciju. Primjerice ako je q = 4, prediktori za x(t) će biti e(t-1) ... e(t-4)
d	Odnosi se na broj promatranih podataka u prošlosti oduzeti s trenutnim vrijednostima. Ovisi o diferenciranju i koji red smo odabrali

Parametri p,q,d se mogu naći koristeći ACF i PACF.



Slika 21. Autokorelacija. Izvor: vlastiti rad

Autokorelacija je mjera korelacije između vremenskog niza sa *zaostajalom* verzijom tog istog niza. Funkcija Autokorelacije nam pokazuje koliko daleko se vremenski niz razlikuje sa svojom prošlosti i koliko su sadašnjost i prošle opservacije povezane. Želimo znati taj podatak, jer želimo identifikator koliko daleko informacija u prošlosti nam može pomoći za predikciju u budućnosti koristeći isti skup.

Na y osi vidimo vrijednosti koje može poprimiti svaka promatrana vrijednost. Ta vrijednost je u intervalu između -1 i 1. Prva promatrana varijabla će uvijek imati vrijednost 1 jer se uspoređuje sa samom sobom, no kasnije se sve više odmičemo kroz niz. U ovom slučaju idemo do 40 vrijednosti u nazad.

Primjerice sa *zaostalom* vrijednosti od 5 ACF bi usporedila vrijednosti t_1, t_2 s podacima $t_1 -_5$ ... $t_2 -_5$ itd.

U statistici kod svake procjene imamo standardnu pogrešku. Plavo područje na grafu predstavlja interval od 95% pouzdanosti odnosno signifikantnosti u odnosu na autokorelaciju.

Kako bi odabrali ispravan model za naš slučaj trebamo uzeti u obzir rezultate mjera koje smo opisali i napravili. To su diferenciranje i autokorelacija. Pri identifikaciji modela valja slijediti sljedeće korake:

1. Pronaći razinu odnosno stupanj *diffrencinga*, d
2. Proučiti uzorke koji se nalaze u ACF i PACF budući da iz njih možemo izvući zaključke o stupnju kod autoregresije i uklanjanje prosjeka.

   Tablica za ACF i PACF vezana za AR(p), MA(q) i ARMA(p,q) procese. Odnose se na brojeve ARIMA modela opisane na početku poglavlja. Prateći sljedeću matricu možemo dobiti svoj izbor.

Process	**ACF**	**PACF**
AR(p)	tails off	cutoff after lag p
MA(q)	cutoff after lag q	tails off
ARMA(p,q)	tails off	tails off

   Za pronalazak optimalnih parametara ARIMA modela grafičke metode vizualizacije nisu najbolje rješenje. Razlog je što problem nije trivijalan i odmah očit, uz to i zahtjeva puno vremena. Za pronalazak najboljih parametara koristit ćemo optimizaciju hiperparametara.

   Ideja je da prođemo i istražimo različite kombinacije parametara. Za svaku kombinaciju probat ćemo prilagoditi novi ARIMA model sa `SARIMAX()` funkcijom iz paketa `statsmodels`, gdje S stoji za Sezonalnost.

   Kada dobijemo cijeli uvid i kombinacije parametara, optimalni set parametara bit će onaj koji ima najbolje rezultate.

```
p = d = q = range(0,2) # Dodjeljujemo p,d,q da mogu primiti bilo koju vrijednost
0 do 2
pdq = list(itertools.product(p, d, q)) # Generiramo sve moguce kombinacije
pdq_x_QDQs = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

for param in pdq:
    for seasonal_param in pdq_x_QDQs:
        try:
            mod = sm.tsa.statespace.SARIMAX(mte,
                                            order = param,
                                            seasonal_order = seasonal_param,
                                            enforce_stationarity = False,
                                            enforce_invertibility = False)
            results = mod.fit()
            print('Arima{}x{} - AIC: {}'.format(param, param_seasonal,
results.aic))
```

```
        except:
            continue

print(results.summary())
```

Zanima nas uglavnom vrijednost AIC (Akaike Information Criterion). AIC procjenjuje koliko dobro model odgovara podacima pritom uzevši u obzir kompleksnost modela. Najmanji AIC znači najbolji model. Najbolji rezultat dao je model SARIMAX(1,1,1) x (0,1,1,12) s AIC vrijednošću od 2003.553.

Možemo konkretizirati na sljedeći način

```
mod = sm.tsa.statespace.SARIMAX(mte,
                                order=(1,1,1),
                                seasonal_order=(0,1,1,12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary())
```

```
                                     Statespace Model Results
==========================================================================================================
Dep. Variable:     Natural Gas Electric Power Sector CO2 Emissions   No. Observations:                  523
Model:                                SARIMAX(1, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -997.777
Date:                                             Tue, 06 Aug 2019   AIC                           2003.553
Time:                                                     09:05:17   BIC                           2020.380
Sample:                                                  01-31-1973   HQIC                          2010.158
                                                       - 07-31-2016
Covariance Type:                                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.6684      0.040     16.560      0.000       0.589       0.748
ma.L1         -0.9534      0.020    -46.799      0.000      -0.993      -0.914
ma.S.L12      -0.7287      0.027    -27.036      0.000      -0.782      -0.676
sigma2         3.2378      0.133     24.263      0.000       2.976       3.499
==========================================================================================
Ljung-Box (Q):                       65.99   Jarque-Bera (JB):               183.59
Prob(Q):                              0.01   Prob(JB):                         0.00
Heteroskedasticity (H):               7.58   Skew:                             0.43
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.85
==========================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

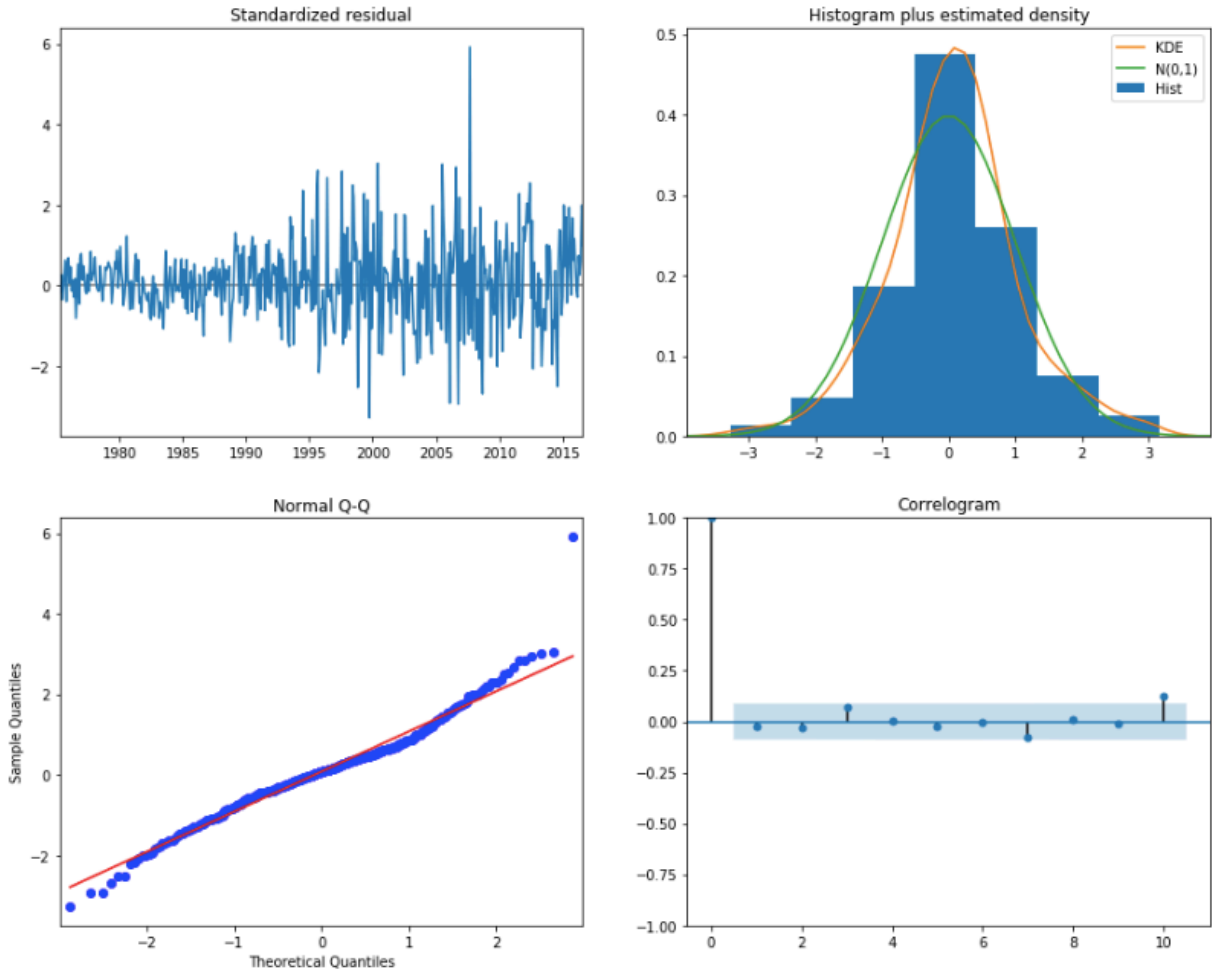
Slika 22. Rezultati SARIMAX metode. Izvor: vlastiti rad

Kolona coef nam prikazuje *weight* odnosno važnost svakog svojstva te kako svaki od njih utječe na vremenski niz. P>|z| kolona upućuje na signifikantnost *weighta* za svakog od njih.

Dijagnostika modela

Pri razvijanju i zaključivanju koji je model najbolji za naš slučaj, potrebno je i napraviti dijagnostiku istog modela. Na taj način se možemo uvjeriti u točnost i sigurnost naših pretpostavki, te da se nikakva svojstva nisu prekršila.

Dijagnostiku radimo na način da vizualiziramo *ostatke*, odnosno niz nakon što uklonimo sezonalnost i trend. Tako možemo provjeriti postoji li još kakav trend koje model nije uhvatio.



Slika 23. Dijagnostika modela. Izvor: vlastiti rad

Kod dijagnostike modela najbitniji pokazatelji su nam da model nema korelaciju u podacima te da je normalno distribuiran između prosječne vrijednosti 0. Ako ti uvjeti nisu zadovoljeni, model se može još poboljšati.

- Gornji lijevi graf pokazuje distribuciju ostataka koji po uzorku liči na bijeli šum. Nema određenog trenda i sezonalnosti.
- Gornji desni graf pokazuje kako je središnja vrijednost normalno distribuirana oko nule, a standardna devijacija 1.
- Q-Q dijagram i autokorelacija potvrđuju da nema sezonalnosti i trenda unutar skupa.

Dijagnostikom možemo zaključiti da je model zadovoljio sve kriterije i možemo krenuti u sam proces predikcije.

4.4 Prediktivni model

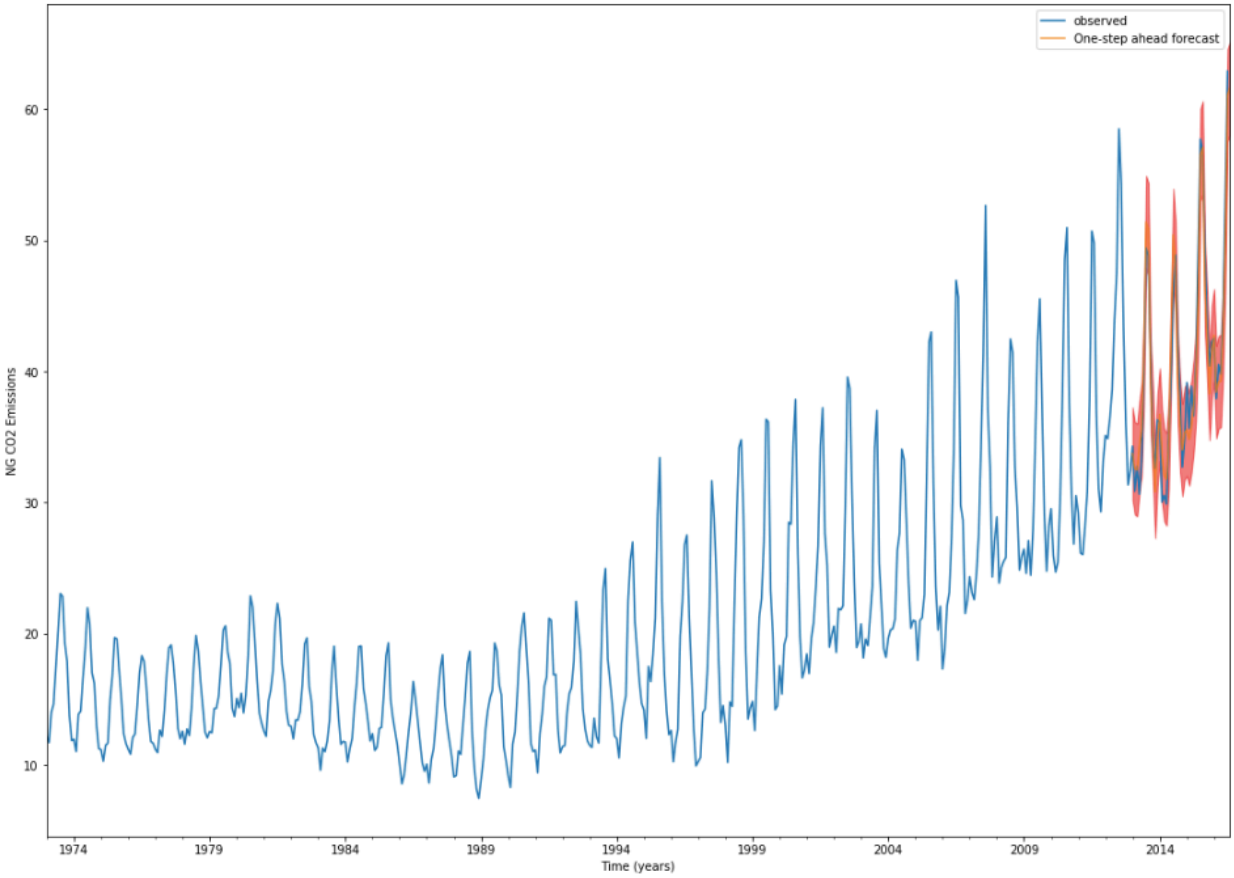
Kako bi provjerili točnost modela možemo usporediti predviđene podatke s onima koji su stvarni i vidjeti koliko su predviđene brojke blizu stvarnih. Inicijalno koristimo metodu "*prvi-korak*"(eng. One step ahead). Dakle uzimamo cijelu povijest vrijednosti unutar vremenskog niza do promatrane točke.

```
ax = mte['1973':].plot(label='observed')
pred.predicted_mean.plot(ax = ax, label='One-step ahead forecast', figsize=(18, 13))

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color = 'r', alpha = .5)

ax.set_xlabel('Time (years)')
ax.set_ylabel('NG CO2 Emissions')

plt.legend()
plt.show()
```



Slika 24. Korak-ispred preditkivni model. Izvor: vlastiti rad

Prema izlaznim rezultatima možemo vidjeti kako nam predviđeni rezultati u odnosu na stvarne stoje poprilično blizu. Prediktivni model uzeo je u obzir trend i sezonalnost na ispavan način.

Kako bi točnost još formalizirali možemo napraviti određene mjere poput MSE koji prikazuje prosječnu pogrešku našeg predviđanja. Za svaku vrijednost koju previđamo on računa udaljenost od prave vrijednosti te kvadriramo rezultat (radi negativnih vrijednosti).

```
mte_forecast = pred.predicted_mean
mte_truth = mte['2013-01-31':]

# Izračun MSE-a
mse = ((mte_forecast - mte_truth) ** 2).mean()
print('The Mean Squared Error (MSE) of the forecast is {}'.format(round(mse, 2)))
```

```
The Mean Squared Error (MSE) of the forecast is 4.09
```

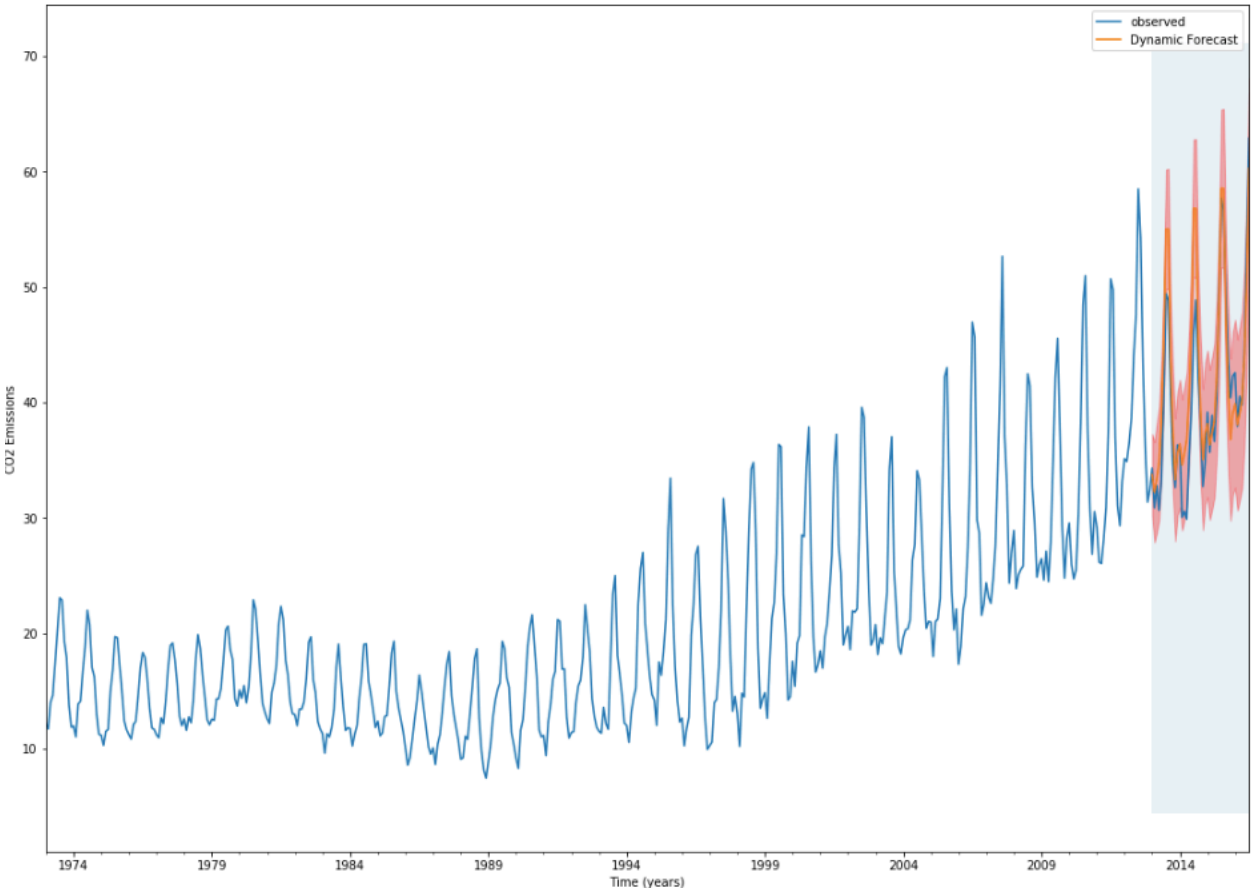
Vidimo da je MSE u vrijednosti 4.09, što je relativno blizu 0. Kada bi MSE imalo vrijednost 0 to bi značilo da je prediktivna vrijednost potpuno točna, to bi naravno bio idealan scenarij no u većini slučajeva nije moguće.

Možemo dok razvijamo model probati malo poboljšati taj MSE kako bi dobili bolju preciznost i kako bi prognozirana vrijednost bila što bliža stvarnoj.

Uz takozvanu "*Korak-ispred metodu*" postoji i ponekad bolja reprezentacija modela. To se može postići tako da koristimo dinamičko prognoziranje gdje uzimamo u obzir informacije vremenskog niza do određene točke u vremenu.

```
pred_dynamic = results.get_prediction(start = pd.to_datetime('2013-01-31'), dynamic =
True, full_results = True)
pred_dynamic_ci = pred_dynamic.conf_int()
```

U gornjem dijelu koda specificirali smo početak izračuna dinamičkog prediktivnog modela od Siječnja 2013 prema naprijed.



Slika 25. Dinamički SARIMAX model. Izvor: vlastiti rad

Pri detaljnijoj procjeni modela vizualna procjena nije baš najbolji pristup. Svakako vidimo da je prediktivna linija leži uz promatranu međutim najbolje bi bilo da napravimo MSE i RMSE kako bi usporedili sa "*korak-ispred*" metodom i dobili konkretne podatke za usporedbu.

```
# Izvlačenje podataka
mte_forecast = pred_dynamic.predicted_mean
mte_orginal = mte['2013-01-31':]

# Izračun MSE-a
mse = ((mte_forecast - mte_orginal) ** 2).mean()
print('The Mean Squared Error (MSE) of the forecast is {}'.format(round(mse, 2)))
```

```
The Mean Squared Error (MSE) of the forecast is 14.39
```

Kod dinamičkog prediktivnog modela MSE je znatno veći. To je očekivan rezultat budući da se oslanjamo na manji broj povijesnih podataka u odnosu na prošli model.

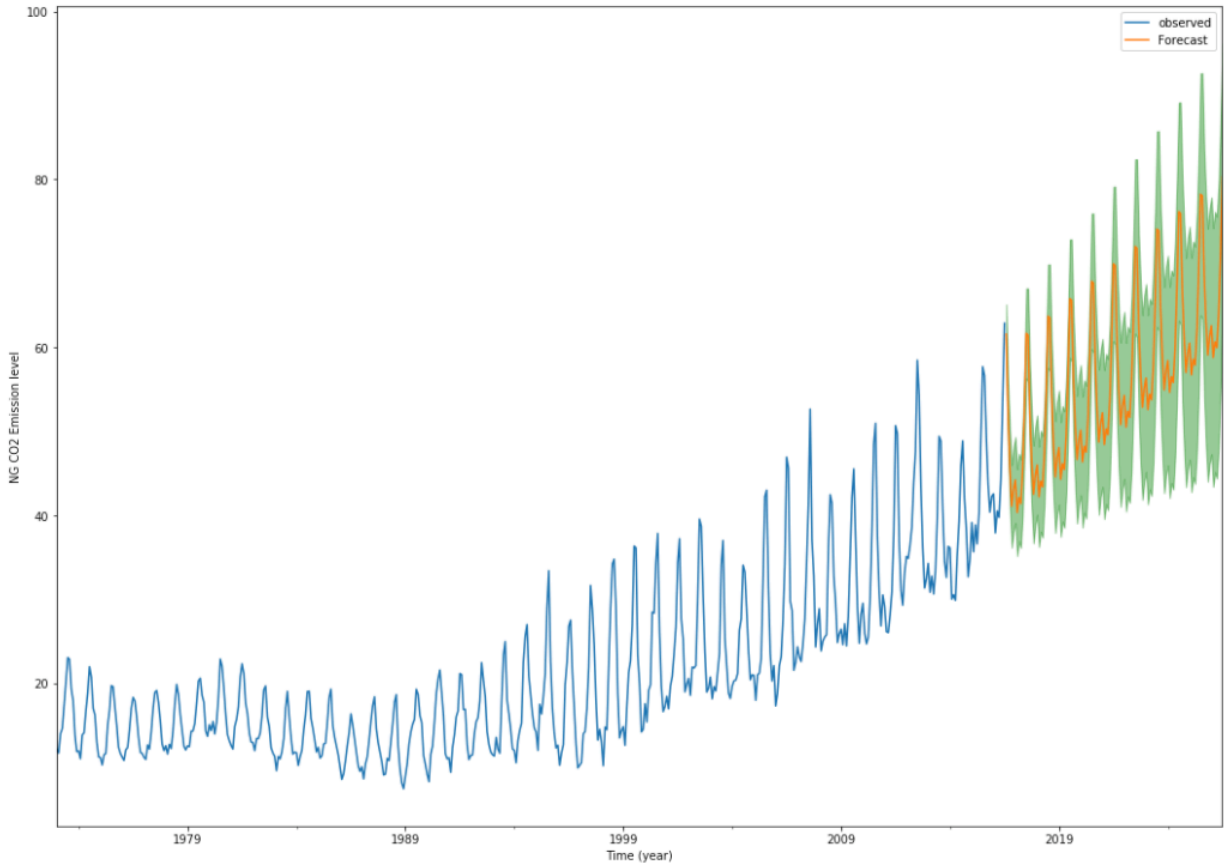
Oba modela mogu proći kao valjana s obzirom na izlazne rezultate. Nakon tog zaključka možemo probati napraviti buduće predikcije. Koristit ćemo funkciju `get_forecast().`

```
# Definiranje koliko ćemo daleko napraviti predikcije, u ovom slucaju 10 godina
forecast = results.get_forecast(steps= 120)

forecast_ci = forecast.conf_int() # Definiranje intervala pouzdanosti
forecast_ci.head()
```

```
ax = mte.plot(label='observed', figsize=(20, 15))
forecast.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(forecast_ci.index,
                forecast_ci.iloc[:, 0],
                forecast_ci.iloc[:, 1], color='g', alpha=.4)
ax.set_xlabel('Time (year)')
ax.set_ylabel('NG CO2 Emission level')

plt.legend()
plt.show()
```



Slika 26. Prognoza SARIMAX modela. Izvor: vlastiti rad

S ovakvim modelom možemo izvući određene zaključke i prognoze. Odmah je vidljivo da se prognozira postupni rast u emisiji ugljikovog dioksida. Svakako da se model može dalje analizirati i mjeriti.

Naravno treba imati na umu da je kako vrijeme ide dalje u budućnost sve teže i neizvjesnije napraviti točno predviđanje te s vremenskim odmakom naš interval pouzdanosti raste. [2]

5. Modeliranje pomoću TensorFlow STS biblioteke

U ovom poglavlju upoznat ćemo i predstaviti Pythonov paket TesnorFlow Structural Time Series koji je integriran unutar TensorFlow Probability razvojnog okvira.

U dokumentaciji paketa on se opisuje kao aplikacijski okvir za strukturalne vremenske nizove koristeći Bayesove principe[44]. Na taj način odmičemo se od klasičnih i standardnih metoda modeliranja vremenskih nizova. Glavne statističke metode koje se koriste u modelima strukturalnog vremenskog niza su tzv. "*state space*" forme modela te Kalman filter.

Modeli strukturalnih vremenskih nizova su u svojoj suštini modeli regresije. Gdje je promatrana varijabla funkcija kroz vrijeme i gdje parametri variraju kroz vremenski period. Takav model se zapravo integrira s "state-space" modelom. Gdje ta druga "state-space" komponenta predstavlja stanje komponenti niza kao što su trend i sezonalnost. Estimacija se ažurira kada nove opservacije postanu dostupne.

5.1 Strukturalni vremenski nizovi

Možemo ponoviti kako se strukturalni vremenski nizovi dekomponiraju se u trend, ciklus, sezonalnost te određeni stupanj greške. Dekomponiranjem svake od tih komponenti dobivamo detaljnije uvide u distribuciju te ih promatramo na izolirani način. Svaka od tih komponenti tada može sadržavati svoju interpretaciju tijekom procesa modeliranja.

Znamo da se vremenski niz predstavlja kao niz opservacija $y_1, \ldots, y_n$ kroz vrijeme. Klasičan način na koji model prikazujemo je aditivan, dakle:

$$y_t = \mu_t + \gamma_t + \varepsilon_t$$

Gdje je μ_t sporo mijenjajuća komponenta zvana trend, γ_t periodička komponenta s fiksnim periodima - sezonalnost te ε_t greška distribucije odnosno šum.

Dekompozicijom naši modeli dobivaju prednost u izradi prognoza upravo radi toga što možemo individualno promatrati pojave kao što su trend i sezonalnost te ih zasebno modelirati.

U prijašnjim poglavljima smo radili s univarijatnom obradu podataka gdje je u određenom vremenu jedna varijabla promatrana.

Multivarijatna obrada podataka može analizirati više varijabli zajedno te promatrati njihove interakcije, povezanost itd. Regresija, stabla odlučivanja i klasteriranje su neki od metoda koji koriste multivarijatnu obradu podataka.

Proći ćemo neke od metoda koji se vežu uz klasične tehnika modeliranja strukturalnih vremenskih nizova. Takve metode su:

- Local level model
- Local linear trend model
- Model sa sezonalnom komponentom
- Model s regresijom

5.2 Local level model

Ovo je najjednostavniji model unutar STS paketa korišten u svrhu prognoziranja vremenskih nizova.

Pojam slučajne šetnje odnosi se na jako varijabilan rast ili pad vremenskog niza. U takvom slučaju bolje je ne pokušavati predvidjeti kretanje niza već izračunati postotak promjene između dva perioda (y_t - y_{t-1}). Svakom daljnjom kretnjom vremenskog niza vrijednost varijable tako uzima nezavisnu varijablu koja predstavlja slučajnu šetnju. [5]

Local level model uzima koncept slučajne šetnje, gdje je zapravo μ_t komponenta sa slučajnom šetnjom. Ona ne sadrži sezonalnost te su slučajne varijable normalno distribuirane.

Sastoji se od sljedećih jednadžbi:

$$y_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{NID}(0, \sigma_\varepsilon^2)$$
$$\mu_{t+1} = \mu_t + \eta_t, \qquad \eta_t \sim \mathcal{NID}(0, \sigma_\eta^2),$$

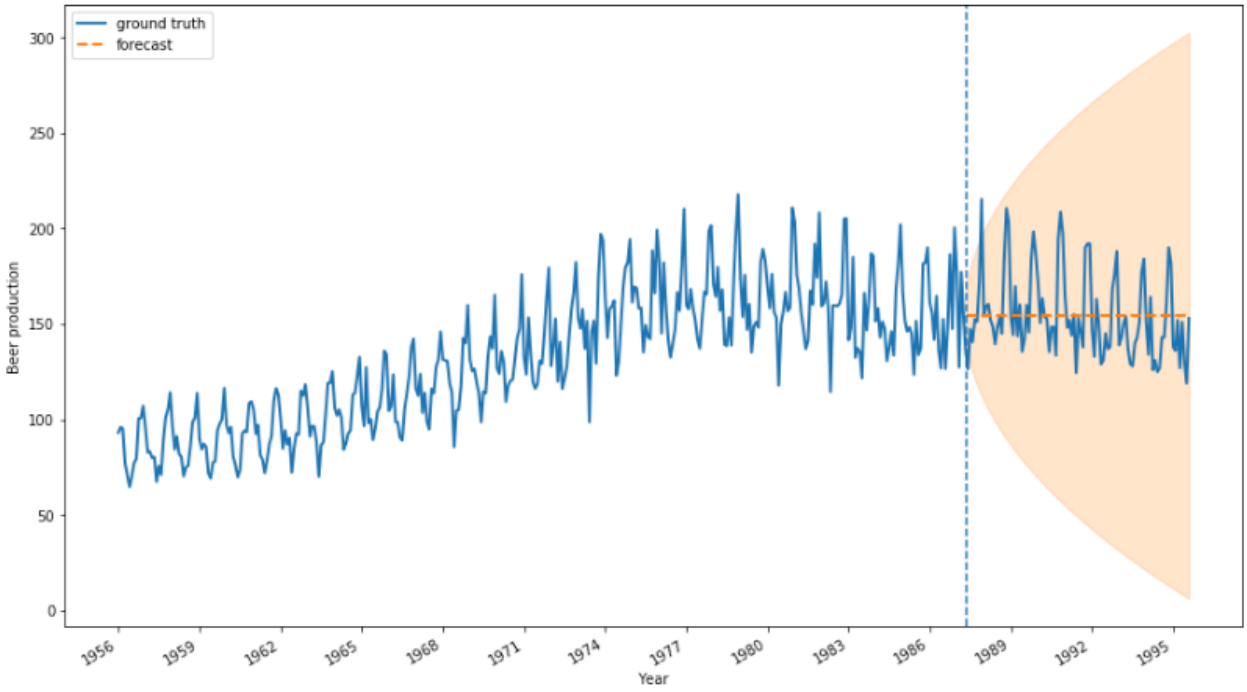
Slika 27. Izvor: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

Prva jednadžba se odnosi na izračun promatranih varijabli gdje je ε greška odnosno šum koji ne možemo objasniti i koji se propagira u obliku varijacije.

Ovo je temeljni model strukturalnih vrmenskih nizova, gdje uz promatrane varijable y_t imamo $\mu_t + 1$ .Druga jednadžba prikazuje niz nepromatranih varijabli $\mu_1, \ldots, \mu_t + 1$ zvano stanje. To stanje predstavlja ponašanje i razvoj niza kojeg proučavamo kroz vrijeme.

```
def build_model(observed_time_series):
    local = sts.LocalLevel(observed_time_series = observed_time_series)
    return local
```

Funkcija za kreiranje prediktivnog modela je poprilično jednostavna, u varijablu spremamo objekt LocalLevel iz klase Structural Time Series paketa te mu kao argument dajemo naš skup podataka.



Slika 28. Lokalni level model. Izvor: vlastiti rad

Možemo vidjeti kako je predikcija uvijek ista kroz vrijeme. To smo i očekivali budući da Local Level model ne uzima u jednadžbu komponente kao što su sezonalnost i trend.

5.3 Local linear trend model

Local linear trend nastavlja se na prethodni osnovni model. Međutim u jednadžbu se ubraja i trend.

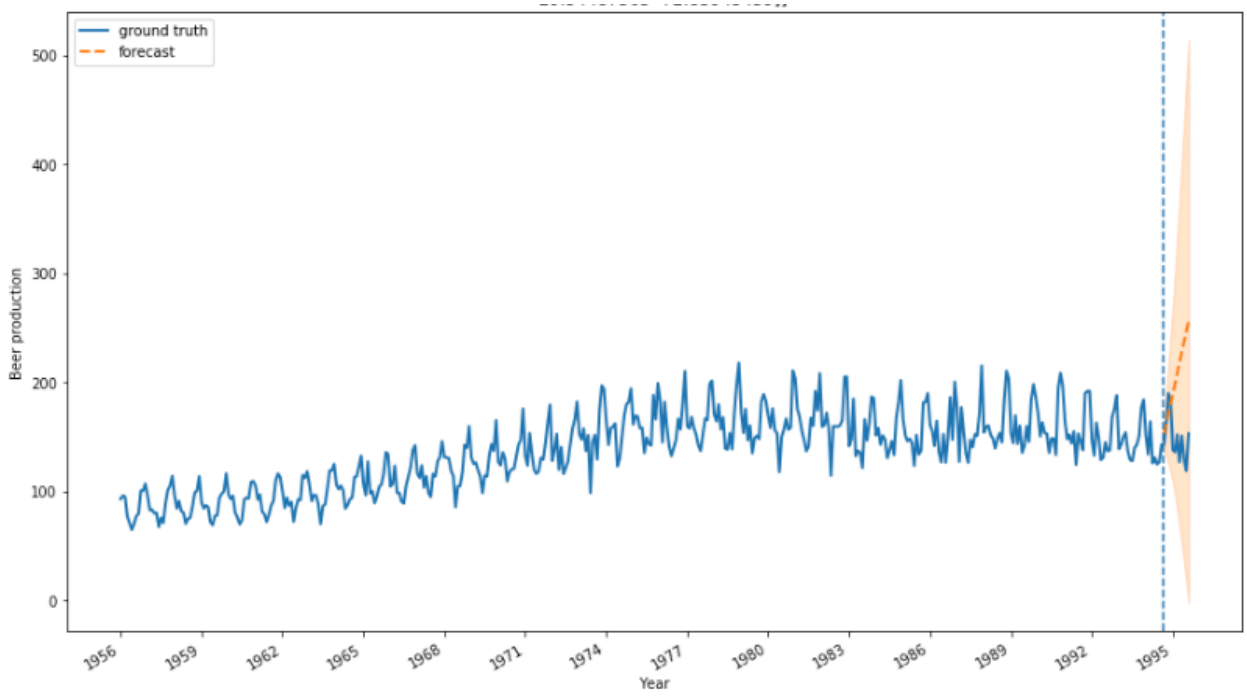
$$
\begin{aligned}
slope_t &= slope_{t-1} + \epsilon_1, & \epsilon_1 &\sim \mathcal{N}(0, \sigma^2_{slope}) \\
level_t &= level_{t-1} + slope_{t-1} + \epsilon_2, & \epsilon_2 &\sim \mathcal{N}(0, \sigma^2_{level}) \\
y_t &= level_t + \epsilon_3, & \epsilon_3 &\sim \mathcal{N}(0, \sigma^2_{obs})
\end{aligned}
$$

Slika 29. Izvor: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

Kod navedenih jednadžbi, u svakom koraku kroz vrijeme t, *slope*, *level* i *y* su slučajne varijable. S tim da treba uzeti u obzir da one ipak ovise o prošlim vrijednostima t-1, dok ε_1, ε_2 i ε_3 su šum i ne ovise o vremenu i prethodnim varijablama. Local linear trend još se naziva i linearan dinamički sustav. Gore tri navedene komponente su parametri modela.

Uz svaku jednadžbu koja pripada ovom modelu veže se i Gaussov šum koji imaju prosječnu vrijednost koja iznosi 0 i standardnu devijaciju pojedine vrijednosti. ε može poprimiti samo pozitivne vrijednosti.

```
def build_model(observed_time_series):
  model = sts.LocalLinearTrend(observed_time_series = observed_time_series)
  return model
```



Slika 30. Local linear trend model. Izvor: vlastiti rad

Vidimo na grafu da model dobro zapaža trend, međutim ne uzima u obzir moguću sezonalnost koja se očigledno pojavljuje u nizu. Prognozirane vrijednosti idu samo prema naprijed na temelju rastućeg trenda.

Prema opisu u dokumentaciji `tensorflow.sts` paketa stoji da je ovakav model prigodan za podatke s jasnim trendom koji je dovoljno konzistentan ali da može evoluirati kroz vrijeme.

Kako bi poboljšali model potrebno je dodati sezonalnost u model.

5. 4 Local linear trend sa sezonalnošću

Budući da se naš vremenski niz sastoji od diskretnog broja sezonalnosti, kako bi što točnije definirali model trebamo uzeti u obzir i tu komponentu niza. Sezonalnost obično predstavlja regularni uzorak unutar vremena kao primjerice dan u tjednu, određeni mjeseci, sati itd. Svaka sezonalnost ima svoje trajanje i efekte na koje utječe. Kao i uz sve jednadžbe i komponente i uz sezonalnost se veže Gaussov šum.

Jednadžba je sljedeća:

$$Seasonal = -\sum \tau_t - _S + w_t$$

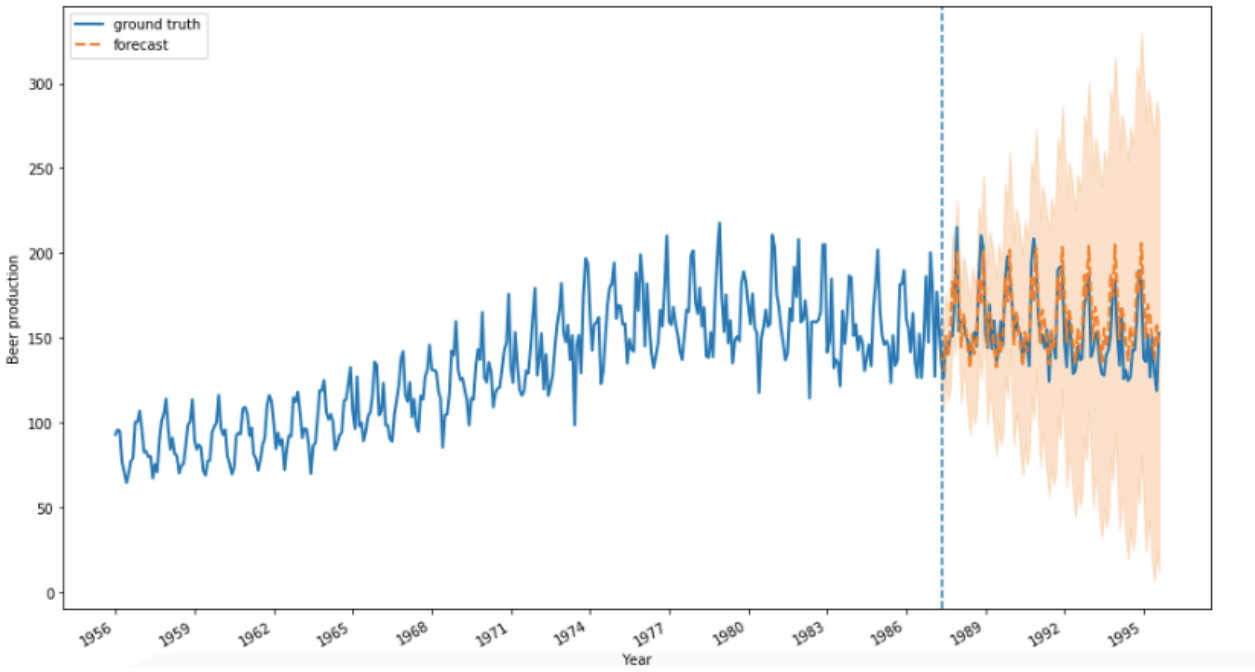
Te jednadžba koja sve ujedinjuje

$$y_t = \mu_t + \tau_t + \varepsilon_t$$

Implementacija u TensorFlow STS paketu:

```
def build_model(observed_time_series):
  trend = sts.LocalLinearTrend(observed_time_series = observed_time_series)
  seasonal = tfp.sts.Seasonal(num_seasons = 12, observed_time_series =
beer_production)
  model = tfp.sts.Sum([trend, seasonal], observed_time_series = beer_production )

  return model
```



Slika 31. Local linear trend sa sezonalnošću. Izvor: vlastiti rad

Možemo vidjeti kako je rezultat neusporedivo bolji u odnosnu na prva dva modela. Naš prediktivni model sada u obzir uzima i trend i sezonalnost i jako dobro predviđa kretanje kroz vremenski niz.

$$x_t = Ax_t - _1 + w_t$$

$$y_t = Hx_t + v_t$$

Ove dvije jednadžbe opisuju linearni trend model kao dinamičan sustav koji se mijenja kroz vrijeme, odnosno evoluira. Gdje je xt varijabla sa stanjem u vremenskom koraku t .

Druga jednadžba yt predstavlja promatranu varijablu. Obje jednadžbe ovise o $xt-1$ varijabli, te je zato sustav linearan. Buduće vrijednosti t se generiraju ovisno o prošlom vremenskom koraku $t-1$.

5.5 Strojno učenje i vjerojatnost

Prije nego krenemo raščlanjivati model i komponente koje ga čine potrebno je pojasniti pojam vjerojatnosne distribucije. Grafički, distribucija vjerojatnosti je krivulja gdje je vjerojatnost određenog ishoda proporcionalna visini te krivulje.

Tipove varijabli u takvoj distribuciji možemo klasificirati u sljedeće kategorije:

- Z- je diskretan - u tom slučaju varijable mogu poprimiti vrijednosti unutar liste. Primjer takve distribucije je populacija, recenzije itd.
- Z je kontinuiran - kod kontinuiranih varijabli raspon je proizvoljan. Primjerice brzina, vrijeme temperatura modeliraju se kao kontinuirane varijable, takve varijable mogu imati višu razinu preciznosti.
- Z kombinacija - varijable poprimaju vrijednosti i od diskretnih i od kontinuiranih.

U ovom poglavlju zanima nas kontinuirana distribucija. Varijable kontinuirane distribucije imaju funkciju gustoće vjerojatnosti. Jedan primjer takve funkcije može biti eksponencijalna slučajna varijabla. [6]

Model je distribucija skrivenih varijabli z i promatranih varijabli x. U ovoj jednadžbi radimo pretpostavke o našem skupu podataka.

$$p(z,x)$$

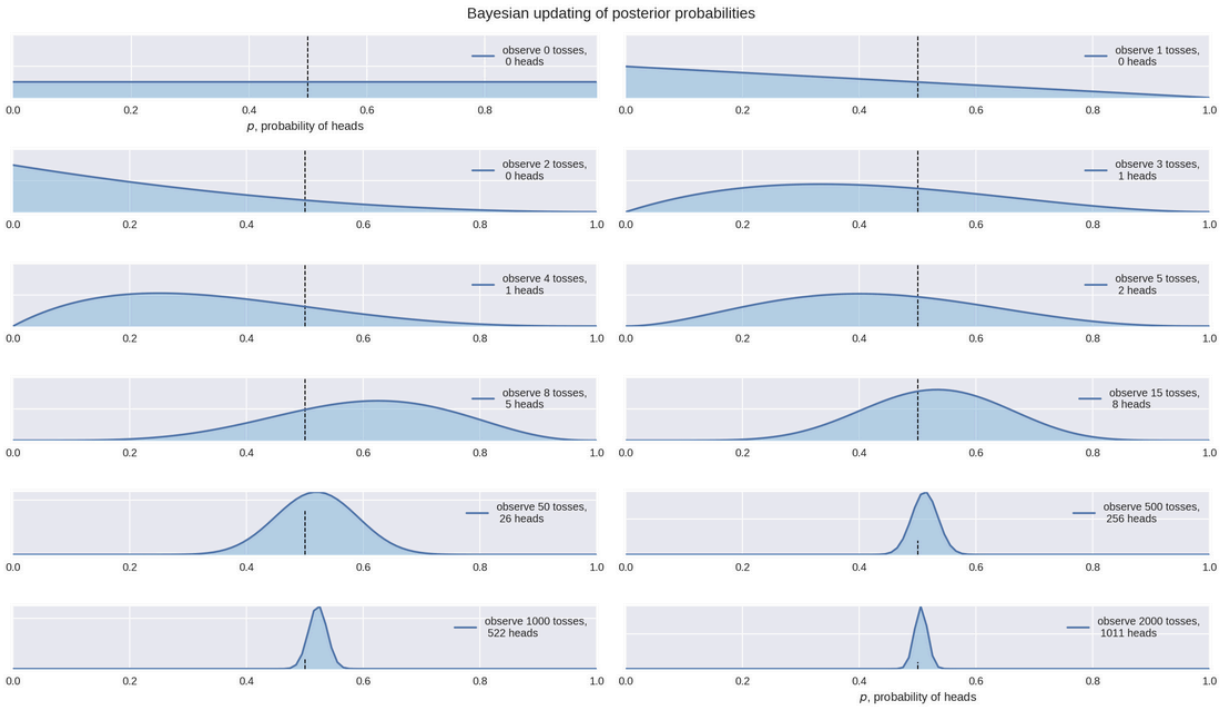
Zaključak o nepoznatom se računa kroz posteriornu vrijednost (distribuciju), kada na promatranu vrijednost dodajemo uvjet. Pri kreiranju modela zamislimo određeni uvjet i skrivene uzorke koji se dešavaju ili ne dešavaju.

$$p(z|x) = \frac{p(z,x)}{p(x)}$$

S ovom jednadžbom zapravo dijelimo udružene distribucije z i x, s vjerojatnošću za koju postavljamo uvjet. Većinom je ovo predmet računanja u sklopu strojnog učenja budući da dobivamo kao vrijednost aproksimaciju.

Želimo što bolje riješiti problem kako da dobijemo $p(z)$ uz dani x gdje x može poprimiti razne pretpostavke u smislu što je z i kakvu oni imaju interakciju.

Vodeći se Bayesovim teoremom želimo izraziti vjerovanje, odnosno vjerojatnost koja se može interpretirati. Tako imamo prior vjerojatnost u događaj A koja je formirana na temelju prošlih informacija. Pritom promatramo naš dokaz i informacije te zatim ažuriramo naše vjerovanje ako ima dokaza za time. [4]



Slika 32. Izvor: https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/blob/master/Chapter1_Introduction/Ch1_Introduction_TFP.ipynb

Na slici možemo vidjeti primjer ponašanja prior i posterior distribucije u slučaju bacanja novčića. Kroz simulacije i promatranje određenih vrijednosti donosimo zaključke o vjerojatnosti pojedinog scenarija. Kroz dovoljan broj ponavljanja vidimo kako se naša posterior vjerojatnost centrira sve bliže vjerojatnosti od *p* = 0.5. Dakle naše vjerovanje obnavljamo na temelju informacija koje dobivamo tokom novih podatka.

Parameter Learning

Prije smo jednadžbama prikazali kako postoje parametri u modelu. Ti parametri kroz vrijeme dobivaju svoju vrijednost. Tensorflow Time Series tretira ovu metodu kao Bayesov model. Tako da na optimiziran način dobivamo parametre za model.[5]

Možemo definirati Y kao vektor svih opservacija u vremenskom nizu. Uz to, možemo definirati z kao vektor parametara modela iz gornje jednadžbe [σ _level, σ _slope, σ _obs].

Posterior distribuciju označavamo kao $p(z|x)$. Ona se računa Bayesovim teoremom:

$$p(z|x) = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

P(z)

$p(z)$ prior vrijednost, naša prvotna pretpostavka o parametrima modela. Unutar local linear trend modela pretpostavljamo da sva tri parametra modela (level, slope, observation) dolaze iz Log Normalne distribucije. Zato što modeliramo standardne devijacije koje uzimaju samo pozitivne vrijednosti.[5]

Također pretpostavljamo da σ _level, σ _slope, σ _obs ne ovise jedni o drugom, stoga da $p(z)$ iznosi:

$$\begin{aligned} p(z) &= p(\sigma\ level, \sigma\ slope\ , \sigma\ obs\ ) \\ &= p(\sigma\ level) * (\sigma\ slope\ ) * (\sigma\ obs\ ) \\ &= LN(\sigma\ level; *)LN(\sigma\ slope; *)LN(\sigma\ obs; *) \end{aligned}$$

Prvom jednadžbom smo raspisali zapravo od čega se sastoji $p(z)$. Zatim smo izrazili nezavisnost između tih varijabli tako da smo ih pomnožili, te nakon toga prikazali kao umnožak od tri individualne Log Normalne distribucije.[5]

P(y|z)

$p(y|z)$ predstavlja vjerojatnost promatranog skupa y uz dani model s parametrima z. Generalna forma takvog modela je:

$$p(y|z) = p(y_1, y_2, \ldots, y_t|z)$$

Model local linear trend takvu formulu malo modificira. Znamo da x_t linearno transformira ovisno o x_t-_1 uz Gaussov šum koji je slučajna varijabla. Stoga i linearna transformacija rezultira Gaussovom varijablom. Isto tako, vjerojatnost promatranog skupa $p(y|z)$ daje Gaussovu slučajnu varijablu.[5]

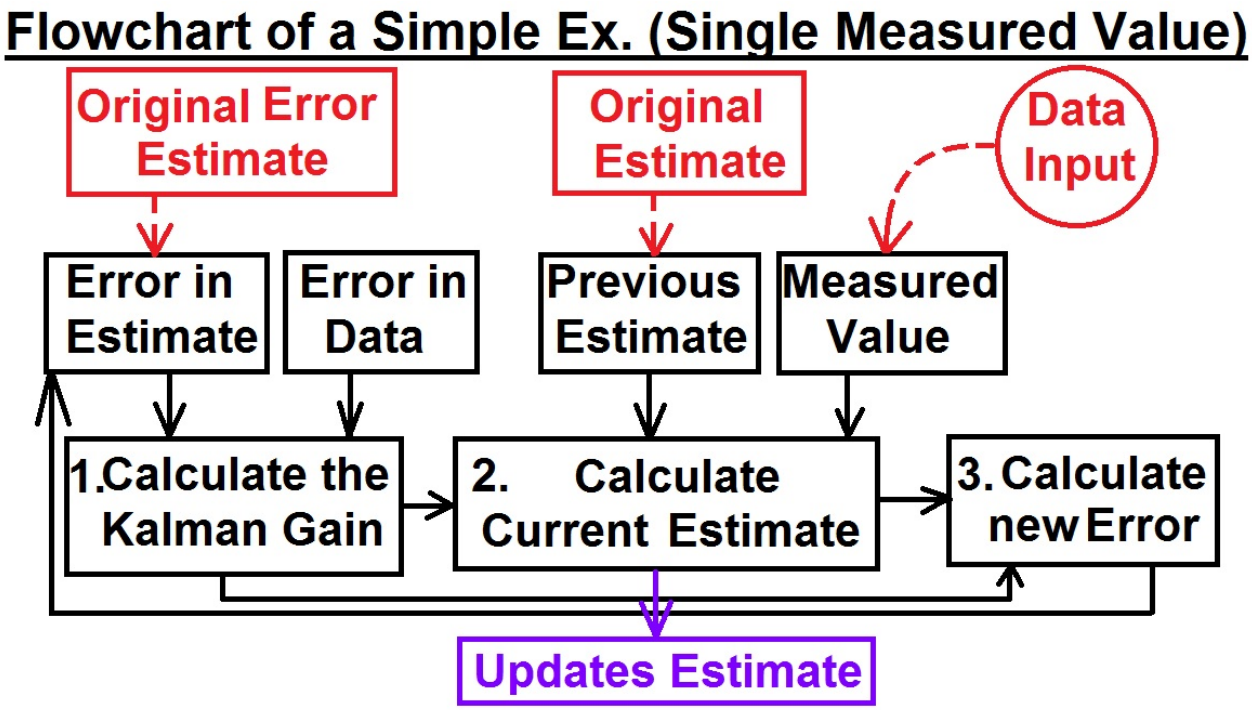
5.6 Kalman Filter

Cilj Kalman filtera jest ažurirati znanje o "*stanju (eng. state)*" varijabli rekurzivno kada novi podaci u distribuciji postanu dostupni [5] . Radi se o iterativnom procesu koji koristi set jednadžbi za izračun procjene prediktivnih vrijednosti, algoritam se ponavlja s dobivanjem novih ulaznih vrijednosti kroz distribuciju i niz.

Metoda Kalman filtriranja ne zahtijeva ogromnu količinu podataka u samom početku već vrlo brzo krene raditi s malim brojem podataka. Zatim procjenjuje pokušava odrediti pravu vrijednost opservacije, odnosno ukloniti šum. Uz to estimira poziciju, brzinu i druge korisne parametre objekta kojeg mjerimo.

Tu se pojavljuje pojam signal — šum. Naši ulazni podaci obilježeni su određenom razinom šuma i slučajnosti, Kalman filter pokušava estimirati i dobiti onu pravu vrijednost te ukloniti šum i varijaciju kako bi imali što bolje predikcije i *pametniji* model.

Intuitivno možemo pretpostaviti da takav algoritam puno brže može odrediti neku centralnu vrijednost distribucije, te da ne moramo čekati veliki skup ulaznih podataka kako bi izračunali središnju vrijednost. Kalman filter nam omogućava bolju i bržu procjenu već u samom startu.



Slika 33. Preuzeto sa YouTube videozapisa The Kalman Filter - Michael Van Biezen (link:)https://www.youtube.com/watch?v=tk3OJjKTDnQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT&index=2

Na slici vidimo dijagram toka za izračun Kalmanovog filtera. Izračun Kalman Gaina je srce ovog algoritma, ta komponenta utječe na ponašanje cijele distribucije.

$$KG = \frac{Eest}{Eest + Emea}$$

Gdje je $Eest$ pogreška u estimaciji, a $Emea$ pogreška u podacima koje promatramo. Broj koji KG može poprimiti kao vrijednost je unutar intervala $0 \leq KG \leq 1$.

Već smo ustanovili da se Kalmanov filter iterativno koristi kako dobivamo nove ulazne vrijednosti. Tako možemo reći da je trenutna estimacija $Eest_t$, a prethodna $Eest(_t-_1)$.

Tako s novim podacima računamo novu vrijednosti:

$$Eest_t = Eest_t\ -_1 + KG[MEA - Eest_t\ -_1]$$

Kroz dovoljan broj iteracija KG će se smanjivati i približavati 0, što znači da dolazimo bliže pravoj vrijednosti. Ako je KG blizu 1 to nam govori da je potrebno ažurirati podatke u estimaciji. [7]

5.7 Varijacijska razlika

Metoda varijacijskog zaključivanja pojavila se kao potreba u strojnom učenju i modelima koji se vode Bayesovom statistikom. Ona ima široku primjenu za aproksimaciju posteriorne distribucije gustoće za razne modele.

TensorFlow STS se oslanja na Bayesove metode gdje posteriorne vrijednosti nije jednostavno izračunati i napraviti aproksimaciju. Cilj varijacijske razlike jest aproksimirati uvjetovanu distribuciju latentnih varijabli uz dane promatrane varijable.[6]

Varijacijsko zaključivanje aproksimira posteriornu vjerojatnost $p(z|y)$ s varijacijskom distribucijom $q(z)$.

$$p(z|x,a) = \frac{p(z,x|a)}{\int p(z,x|a)}$$

Opservacije: $x = x_{1:n}$

Skrivene (latentne varijable): $z = z_{1:m}$

Fiksni parametri: a

Cilj je dobiti posterior distribuciju tako da kreiramo varijacijsku distribuciju nad skrivenim, latentnim varijablama.

$$q(z_{1:m}|v)$$

Želimo podesiti postavke v tako da distribucija q bude blizu posterior distribucije. TensorFlow STS paket koristi metodu *mean-field* kako bi dobili distribuciju $q(z)$. U *mean-field* metodi latentne varijable su nezavisne, a svaka od njih ima svoj varijacijski faktor. Varijacijska distribucija se tako raščlanjuje kroz N ulaznih podataka s parametrima koji se međusobne ne dijele. [5]

Kako bi postupno definirali $q(z)$ potrebno je odabrati gustoću vjerojatnosti za svaku varijablu u z.

$$
\begin{aligned}
& q(z) && \text{(1) q(z) notation} \\
&= q(\sigma_{level}, \sigma_{slope}, \sigma_{obs}) && \text{(2) expand z} \\
&= p(\sigma_{level}) \cdot p(\sigma_{slope}) \cdot p(\sigma_{obs}) && \text{(3) mean-field definition} \\
&= \mathcal{N}(\sigma_{level};\, \mu_l, \sigma_l^2) \cdot \mathcal{N}(\sigma_{slope};\, \mu_s, \sigma_s^2) \cdot \mathcal{N}(\sigma_{obs};\, \mu_o, \sigma_o^2) && \text{(4) Individual density} \\
&\doteq \lambda z, vp\,.\, q(z; vp) && \text{(5) API}
\end{aligned}
$$

Slika 34 Preuzeto sa: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

Treba razjasniti da je u zadnjoj formuli z parametar našeg local linear trend modela, dok je vp parametar gustoće vjerojatnosti. On je varijacijski parametar te kontrolira distribuciju $q(z)$.

Mean-field definira varijable kao nezavisne te smo to prikazali umnoškom.

Nakon toga kreiramo gustoću vjerojatnosti za svaku varijablu uz zasebne parametre koja svaka od njih donosi te predstavljaju 3 zasebne Gaussove distribucije. Ti parametri su varijacijski parametri, dok smo prije naveli parametre modela.

Varijable u $q(z)$ tako vračaju Gaussovu distribuciju i imaju raspon mogućih vrijednosti realnih brojeva. Ono što treba imati na umu jest kakvu vrijednost vraća posterior distribucija $p(z|y)$? Možemo vidjeti iz gornje formule da će biti istog tipa kao prior(z) što je zapravo produkt Log Normalne distribucije i raspon vrijednosti pozitivnih brojeva. [5]

Iz tog razloga i ne podudarnosti javlja se potreba za metodom koja se zove transformacija gustoće vjerojatnosti.

5.8 Transformacija gustoće vjerojatnosti

Cilj nam je s ovom metodom transformirati u novu distribuciju $p(u|y)$ koja će poprimati sve realne brojeve. Tako bi posterior distribucija imala jednaki tip ulaznih vrijednosti kao i varijacijska distribucija (obje distribucije su vezane za gustoću).

S obzirom na to da se radi o međusobno nezavisnim varijablama tako ih možemo promatrati prilikom transformacije.

Koristimo eksponencijalnu funkciju za transformaciju. Ona sadrži svojstva koja su bitna pri izgradnji modela, tu se misli na mapiranje jedan prema jedan.

$$\sigma_{level} = e^{u_{level}}$$
$$\sigma_{slope} = e^{u_{slope}}$$
$$\sigma_{obs} = e^{u_{obs}}$$

Slika 35. Preuzeto sa: Preuzeto sa: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

Ovom tehnikom pojednostavili smo problem te originalnu vrijednost varijabli zamijenili eksponencijalnom funkcijom tako da izlazna vrijednost odgovara našem slučaju. Još se samo trebamo pobrinuti kako izgleda posteriorna gustoća vjerojatnosti nakon promjene varijable.

$$\int p(y|\sigma level, \sigma slope, \sigma obs) p(\sigma level, \sigma slope, \sigma obs) d\sigma level$$

Imamo definirani integral, budući da se radi o distribuciji s gustoćom vjerojatnosti, te 3 glavna parametra.

$$\int_0^{+\infty} p(y|\sigma_{level}, \sigma_{slope}, \sigma_{obs}) p(\sigma_{level}, \sigma_{slope}, \sigma_{obs}) \mathrm{d}\sigma_{level} \quad \text{(1) definition}$$

$$= \int_0^{+\infty} p(y|\sigma_{level}, \cdot) p(\sigma_{level}, \cdot) \mathrm{d}\sigma_{level} \quad \text{(2) simplify}$$

$$= \int_{\log(0)}^{\log(+\infty)} p(y|e^{u_{level}}, \cdot) p(e^{u_{level}}, \cdot) \frac{\mathrm{d}e^{u_{level}}}{\mathrm{d}u_{level}} \mathrm{d}u_{level} \quad \text{(3) integration by substitution}$$

$$= \int_{-\infty}^{+\infty} p(y|e^{u_{level}}, \cdot) p(e^{u_{level}}, \cdot) e^{u_{level}} \mathrm{d}u_{level} \quad \text{(4) reduce limits and derivative}$$

Slika 36. Preuzeto sa: Preuzeto sa: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

U 2. koraku pojednostavili smo formulu da konkretiziramo primjer transformacije na jednoj komponenti, u ovom slučaju level.

U 3. koraku primijenili smo pravilo integracije koristeći supstituciju — zamijenili smo originalnu varijablu novom transformiranom funkcijom prikazanu u gornjem primjeru. Zatim smo pomnožili originalnu funkciju s deriviranom transformiranom funkcijom.

Na kraju smo pripazili da pomoću log funkcije napravimo inverz eksponencijalne funkcije kako bi imala monoton rast, odnosno mapiranje jedan na jedan kao bitno svojstvo. Možemo primijetiti kako su se granice intervala promijenile te su inverzne.

5.9 Kako mjeriti sličnost dviju distribucija?

Varijacijska distribucija q(z) zahtjeva da bude što sličnija distribuciji *p(z|y)*. Kada bi to vizualno prikazali da se zapravo preklapaju površinski što je točnije i bolje moguće.

Oblik q(z) distribucije kontroliran je s parametrima koje smo naveli u gornjem poglavlju. Ono što želimo saznati su vrijednosti parametara modela z. Promatrat ćemo njegove vrijednosti tako da se q(z) preklapa s posterior distribucijom p(z|y) što je više moguće. Za to moramo imati nekakvu "definiciju" sličnosti, koju definiramo Kullback-Leiblerrovom razlikom. Ona može poprimiti pozitivne vrijednosti ili biti 0.

Promatrajmo naše opservacije odnosno skup podataka kao X, te Z kao latentne (skrivene) varijable. Latentne varijable su kreirane iz prior distribucije $p(z)$. Sami podaci x imaju vjerojatnost $p(x|z)$ koja je uvjetovana latentnim varijablama z.

Naš model predstavlja vjerojatnosnu distribuciju $p(x, z)$ gdje su podaci i latentne varijable međusobno zavisne. Takvu vezu možemo dekomponirati u prije spomenutu vjerojatnost i prior distribuciju $p(x, z) = p(x|z)p(z)$.

Sada možemo razmišljati o razlici. Cilj nam je dobiti dobre vrijednosti latentnih varijabli uz dani skup podataka, odnosno izračunati posteriornu vrijednost $p(z|x)$. Vodeći se Bayesovim teoremom možemo zapisati sljedeće:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Nazivnik $p(x)$ predstavlja dokaz, te se računa marginaliziranjem latentnih varijabli: $p(x) = \int p(x \mid z)p(z)dz$. Budući da moramo evaluirati integral kroz cijelu konfiguraciju latentnih varijabli trebamo izračunati i posterior distribuciju. Metoda varijacijske razlike aproksimira posterior pomoću tzv. "*obitelji*" distribucija. Varijacijski parametar λ predstavlja obitelj distribucija. Primjerice kada bi imali distribuciju u Gaussovoj formi takav varijacijski parametar bi se mogao izraziti kao: $\lambda x_i = (\mu x_i, \sigma(x_i)^2))$.

Mjerit ćemo razliku dviju distribucija koristeći Kullback-Leibler razliku koja mjeri "izgubljene" informacije koristeći q kako bi aproksimirali p. Ona je definirana sljedećom jednadžbom:

$$KL(q(z)||p(z|y)) = \mathbb{E}_{z\sim q(z)}\left[log\frac{q(z)}{p(z|y)}\right]$$

$$= \int q(z)log\frac{q(z)}{p(z|y)}\mathrm{d}z$$

Slika 37. Elbo Preuzeto sa: Preuzeto sa: https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a

Cilj nam je pronaći varijacijske parametre λ koji su optimum; odnosno minimiziraju razliku.

Kada dobijemo željeni rezultat, tada distribuciju možemo koristiti da dobijemo vrijednosti od z. Želimo pronaći određene vrijednosti za λ tako da $q(z)$ minimizira udaljenost od posteriorne distribucije $p(z|y)$. Sljedeća formula se odnosi na minimizaciju:

$$q_\lambda*(z \mid x) = argmin_\lambda \ KL(q_\lambda \ (z \mid x) \mid\mid p(z \mid x)).$$

Zvjezdica (*) ovdje predstavlja optimalnu vrijednost za $q(z)$. Poznati izraz za ovakvo rješenje naziva se i "node-splitting", gdje u nizu dobivenih rješenja želimo ono s najboljim parametrima — optimumom. Dakle želimo pronaći kombinaciju naših šest varijacijskih parametara tako da KL-razlika bude najmanja.

Navedenu distribuciju nije moguće izračunati direktno, $p(x)$ odnosno dokaz nalazi se u razlici te se ne može pratiti. Kako bi dobili i to svojstvo potrebno je dodati još nešto:

$$ELBO(\lambda) = E\,q\ [logp(x, z)] - E\,q\ [logq\,\lambda\ (z \mid x)].$$

Spomenuli smo da Kullback-Leibler razlika poprima vrijednosti veće ili jednake od nule. To znači da je minimiziranje Kullback-Leibler razlike jednako maksimiziranju ELBO jednadžbe. Vodeći se time ELBO metoda omogućuje nam aproksimiranje posteriorne razlike. Budući da ne možemo pratiti i minimizirati direktno Kullback-Leibler razliku možemo maksimizirati ELBO kojeg možemo u potpunosti pratiti i mjeriti.

ELBO možemo dekomponirati u sumu gdje svaki uvjet ovisi o jednom podatku u nizu. To svojstvo dopušta nam korištenje stohastičkog Gradient Descent uzevši u obzir parametar λ.

Takvu perspektivu ELBO metode, koja promatra svaki pojedini ulaz podataka možemo prikazati na sljedeći način:

$$ELBO_i(\lambda) = Eq\,\lambda\ (z \mid x_i\ )[logp(x_i\ \mid z)] - KL(q\,\lambda\ (z \mid x_i\ )\ ||\ p(z))$$

Iz jednadžbe možemo vidjeti intuiciju ELBO metode, imamo ulazne podatke x_i uz parametar λ. Prvi uvjet koji očekujemo da se zadovolji jest log vjerojatnost koji nam opisuje koliko se dobro naš model uklapa u promatrani skup[2]. Drugi uvjet u jednadžbi jest sama Kullback-Leibler razlika. On opisuje kolika je razlika između naše predložene distribucije $q(z)$ s obzirom na prior distribuciju $p(z)$.

Prvi dio jednadžbe želimo da je što veći s obzirom na to da opisuje vjerojatnost događaja koju želimo da bude što veća. S druge strane želimo da Kullback-Leibler razlika bude što bliže 0.

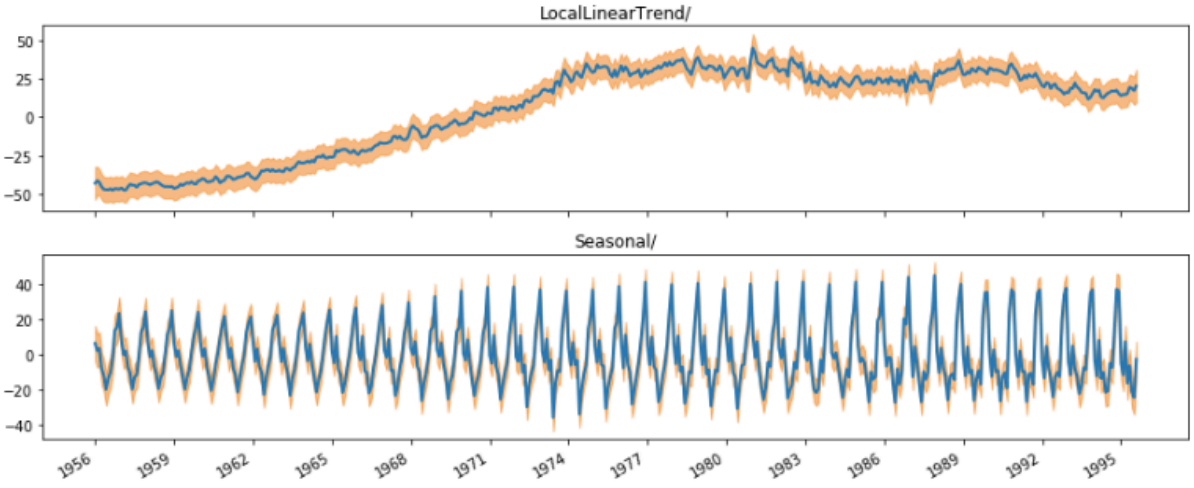
Uz varijacijske parametre, važno je napomenuti kako se u svrhu maksimiziranja ELBO-a i minimiziranja razlike između distribucija može uzeti i u obzir parametri modela. Najčešće su ti parametri *weight* i određene "*predrasude*" neuronske mreže parametrizirajući vjerojatnost događaja uzimajući u obzir dokaze. Takva metoda zove se Varijacijska očekivana maksimizacija (eng Variational expectation maximization)[11].

```
Use tf.where in 2.0, which has the same broadcast rule as np.where
step 0 -ELBO 575.0074362462475
step 20 -ELBO 539.4096038813882
step 40 -ELBO 534.964979486791
step 60 -ELBO 534.4798646666397
step 80 -ELBO 534.6954835222722
step 100 -ELBO 533.2599421940159
step 120 -ELBO 534.8096547141128
step 140 -ELBO 536.4964228055727
step 160 -ELBO 534.0912730636586
step 180 -ELBO 534.503517985024
Inferred parameters:
level_scale: 1.6890013445264345 +- 0.10759792535632969
slope_scale: 1.7768391300609674 +- 0.16065415257308976
```

U gornjem primjeru možemo vidjeti izlazne podatke koji su generirani prilikom korištenja ELBO metode za primjer lokalnog linearnog trenda sa sezonalnošću prikazan na slici 3.

Dekompozicijom i prikazom pojedinih komponenta niza možemo dublje razumjeti naš model. Pomoću TensorFlow STS paketa možemo na intuitivan način vizualizirati i izdvojiti komponente modela — trend i sezonalnost.

```
component_dists = sts.decompose_by_component(
    model,
    observed_time_series=beer_production,
    parameter_samples=q_samples)
```



Slika 38. Dekompozicija sturkturalnog vrmenskog niza. Izvor: vlastiti rad

Uviđamo jaku sezonalnost koja se ponavlja kroz godine, te trend koji ima tendenciju pozitivnog rasta.

6. Zaključak

U radu su pojašnjene razne metode i tehnike modeliranja vremenskih nizova te zatim i prikaz prediktivnih modela i rezultata koje su svaka od njih ostvarile. Pri samom uvodu i upoznavanju vremenskih nizova važno je primijetiti stvari koje treba imati na umu pri modeliranju. Vremenski nizovi zahtijevaju dobro razumijevanje podataka koje promatramo te dekompoziciju dijelova koje ga čine. Vremenski nizovi su specifični zato što se većinom radi o jednoj promatranoj varijabli, koja je u prizmi određenog vremenskog intervala. Uz to traži i razumijevanje podataka kroz vrijeme, te dobru detekciju pojava kao što su sezonalnost i trend.

Prikazani su postupci od samog početka izgradnje modela kao što su čišćenje podatka, testiranje stacionarnosti, vizualizacija i slično. Prošli smo metode eksponencijalnog izglađivanja, ARIMA modele te TensorFlow paket Structural Time Series. U radu je prikazano kako svaki od tih metoda može ostvariti jako dobre rezultate. TensorFlow STS ipak se razlikuje od ostalih metoda budući da se radi o novoj biblioteci koja objedinjuje tradicionalne statističke metode koje se vežu za vremenske nizove i razne algoritme i principe strojnog učenja. Isto tako uzima za perspektivu Bayesovo razmišljanje o ažuriranju dokaza s dolaskom novih informacija.

Svakako treba imati na umu da su biblioteke i razni alati dostupni, međutim od ključne je važnosti razumijevanje samih podataka te osnovnih načela i principa vremenskih nizova. Podaci nose sa sobom svoj kontekst i informacije koje mogu biti jako specifične po tipu i ponašanju. Potrebno je imati znanje iz promatrane domene te znanje o vremenskim nizovima kako bi poduzeli niz mjera kao što su

dekompozicija, autokorelacija i slične te dobili uvid u stanje i prirodu podataka. U takvim raznim slučajevima najbolji model može uvelike varirati.

Korišteno je nekoliko skupova podataka, vremenski nizovi imaju široku primjenu i mogu se koristiti za gotovo sva područja u društvu. Ono što bi bio korak dalje je bilo vidjeti kako bi se mogla ovakva analiza uklopiti u nekakve procese poduzeća u samoj produkciji i razvoju. Kako bi najbolje mogli koristiti modeliranje vremenskih nizova kao potpora odlučivanju i općenito izvlačenju znanja iz poduzeća.

LITERATURA:

[1] Robert Nau, Principles and risks of forecasting, Fuqua School of Business, Duke University September 2014

[2] Rob J Hyndman i George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[3] Dokumentacija TensorFlow Probability biblioteke [https://www.tensorflow.org/probability]

[4] Wei Yi, Demystifying Tensorflow Time Series: Local Linear Trend [https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a]

[5] Cameron Davison Pilon, Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference (Addison-Wesley Data & Analytics) 2015

[6] Variational Inference: A Review for Statisticians, David M. Blei Department of Computer Science and Statistics Columbia University 2018

[GITHUB repozitoriji]:

Structural Time Series Modeling Case Studies: Atmospheric CO2 and Electricity Demand (https://github.com/tensorflow/probability/blob/master/tensorflow_probability/examples/jupyter_notebooks/Structural_Time_Series_Modeling_Case_Studies_Atmospheric_CO2_and_Electricity_Demand.ipynb)