Juha Holstila

Harjoitustyö

Sisällys

1. Yleistä

Harjoitustyönä tutkin onko mahdollista ennustaa NHL-maalivahtien palkkaa perustuen heidän tilastoihinsa pelatuista peleistä.

2. Materiaali

Tässä dokumentissa kuvataan harjoitustyön sisältö. Itse koodi on omasa jupyter-työkirjassa githubissa https://github.com/holezekki99/JODA2022 . Linkki toteutuksen streamlit applikaatioon on samassa paikassa.

3. Harjoitustyön eteneminen

3.1 Kehitysympäristö

Kehitysympäristönä oli pääosin google-colab. Tämä valikoitui helppokäyttöisyyden takia ja sen helppo integrointi github-säilööni. Lopullisen käyttäjäsovellus on toteutettu omalle koneelle pystytetyllä anaconda-ympäristöllä johon data-, ja malli on siirretty.

3.2 Datan kerääminen

Datalähteinä työhön tarvittiin a) maalivahtien tilastot pelatuista peleistä ja b) maalivahtien palkkatiedot.

Tilastot ladataan suoraan moneypuck.com sivustolta. Käytettävissä olisi ollut vanhempiakin tilastoja, mutta koska palkkatiedot olivat saatavissa vain kahden viimeisen kauden osalta niin tilastotkin haettiin vain kausilta 2020-2021 ja 2021-2022.

In [ ]:
```
years = ['2020', '2021']
```

In [ ]:
```
# Get the data for seasons starting 2019-2021 and write them to files
import pandas as pd
import requests

for i in years:
  year = str(i)
  url = ('https://moneypuck.com/moneypuck/playerData/seasonSummary/' + year + '/regular/goalies.csv')
  page = requests.get(url)
  r = requests.get(url)
  fname = ('./data/stat' + year + '.csv')
  open(fname, 'wb').write(r.content)
```

Palkkatiedot hankittiin raapimalla www.spotrac.com sivustolta käyttäen beautifulsoup4-kirjastoa:

In [ ]:

```
from bs4 import BeautifulSoup

# Defining of the dataframe, we collect only name, year and caphit
df = pd.DataFrame(columns=['name', 'season', 'caphit'])

for i in years:
  year = str(i)
  url = ('https://www.spotrac.com/nhl/positional/' + year + '/goaltender/active-cap/')
  page = requests.get(url)
  soup = BeautifulSoup(page.content, 'html.parser')
  table = soup.find_all('table')[1]
  #table = soup.find_all('table')[1] # Table 1 is the list of goalies
  # Collecting Ddata
  for row in table.tbody.find_all('tr'):
    # Find all data for each column
    columns = row.find_all('td')
    if(columns != []):
      name = columns[2].text.strip()
      # Use only familyname since the first names might be spelled differently.
      #name = name.split()[-1]

      # We later found out that there are four names wrongly spelled in salary dataset. Let's correct those:
      # Correct Vasilevskiy familyname
      if name == 'Andrei Vasilevski':
        name = 'Andrei Vasilevskiy'
      # Correct Grubauer's firstname
      if name == 'Phillip Grubauer':
        name = 'Philipp Grubauer'
      # Correct Talbot's firstname
      if name == 'Cameron Talbot':
        name = 'Cam Talbot'
      # Correct Georgievs's firstname
      if name == 'Alexander Georgiev':
        name = 'Alexandar Georgiev'


      year = year
      caphit = columns[3].text.strip()
      df = df.append({'name': name,  'season': year, 'caphit': caphit}, ignore_index=True)
```

Työn edetessä huomattiin, että osa maalivahtien nimistä oli kirjoitettu eri lailla eri paikoissa, joten tärkeimmät niistä korjattiin yhteneviksi yllä.

Lopputulemana oli tiedosto *salary.csv, stats20.csv* ja *stats21.csv*, jotka tallennettiin colabin *data*-hakemistoon.

3.3 Datan jalostaminen

Vaikka data oli aika laadukasta, niin sen jalostamiseen meni paljon aikaa.

Ensin yhdistettiin tilastotiedostot yhdeksi pandasin dataframeksi ja datan tutkiminen alkoi. Selvisi, että joka maalivahdilla oli joka kaudelle oma rivinsä eri pelitilanteille (5 vs.5, 4 vs. 5. 5 vs. 4 jne). Päädyin karsimaan nämä ja ottamaan mukaan vain rivin *all*, joka piti sisällään kaikki pelitilanteet. Tätä olisi ehkä voinut miettiä tarkemmin, koska monasti maalivahtien tietyt ominaisuudet voivat paljastua eri pelitilanteissa. Esim. 4 vs. 5 alivoimassa saattaaisi korostua maalivahdin nopea liikkuminen sivuttain ja 5 vs. 4 ylivoimassa korostuu kyky torjua läpiajoja. Nämä ovat helposti mieleenjääviä ominaisuuksia jotka voivat vaikuttaa kun mietitään tulevaa palkkaa.

Jäähyt poistettiin piirteistä selvästi merkityksettöminä. Muutoin haluttiin säilyttää mahdollisimman paljon piirteitä ja tutkia myöhemmin niiden mahdollista vaikutusta palkkaukseen.

Palkkatilastot ladattiin omaan dataframeen ja tehtiin tietotyyppien vaihto oikeaksi:

```
[128] # Convert year into numeric value:
      sal_df['season'] = sal_df['season'].astype(int)

      # Convert caphit into numeric value.
      sal_df['caphit'] = sal_df['caphit'].replace("[$,]", "", regex=True).astype(int)
```

Lopuksi dataframet yhdistettiin:

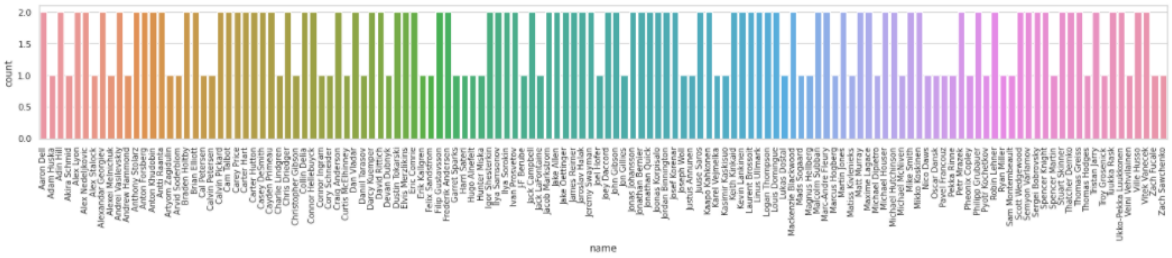
```
[132] merged_df = pd.merge(stat_df, sal_df, how="left", on=["season", "name"])
```

Jatkettiin datan tutkimista. Ensin katsottiin maalivahtien pelattuja kausia seaborn-kirjaston avulla:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()
sns.set(rc={'figure.figsize':(25,3)})
sns.set_style("whitegrid")
sns.countplot(x = 'name', data = df1)

plt.xticks(rotation=90, fontsize=10)
plt.show()
```



NHL:ssä useat maalivahdit pelaavat alemmissa sarjoissa ja heidät kutsutaan NHL:ään vain toedellisen hädän tullen. Haluttiin poistaa nämä tilastosta. Pidettiin vain maalivahdit, jotka ovat pelanneet yli 5 ottelua kauden aikana:

```
[135] df = merged_df
      df = df[df.groupby('name')['games_played'].transform('max') > 5]
      df.shape

      (177, 35)
```

Seuraavaksi alettiin tutkia monta NaN-arvoa meillä on yhdistetyssä dataframessa. Näitä löytyikin aika monta joka johtui siitä, että kaikki maalivahdit eivät ole pelanneet molempana tarkkailukautena. Listasta löytyi monta nimeä, jotka olivat tulleet liigaan vasta kaudelle 21-22 ja kuitenkin vakiinnuttaneet paikkansa. Näiden osalta korvattiin NaN arvo laskemalla keskiarvo pelatuista kausista (huom. datasetti piti aluksi sisällään kolme kautta, mutta kausi 2019-2020 siirtyi työn edetessä maksumuurin taakse. Tästä johtuen laskettiin keskiarvo.):

```
# fill caphit nan values (had not played for that season) with players average caphit
# (https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group)
df['caphit'] = df['caphit'].fillna(df.groupby('name')['caphit'].transform('mean'))
```

Dataan jäi vielä pelaajia, joilla palkka (caphit) oli NaN. Tutkittiin ketä nämä olivat ja päädyttiin poistamaan ne:

```
nan_values = df[df['caphit'].isna()]
nan_values['name'].unique
```

```
<bound method Series.unique of 5             Devan Dubnyk
12          Stuart Skinner
17             Matt Murray
21         Calvin Petersen
32     Mackenzie Blackwood
52            Joey Daccord
67              Dan Vladar
73      Michael Hutchinson
111         Pavel Francouz
115              Nico Daws
123     Michael Hutchinson
129           Akira Schmid
136             Dan Vladar
143       Sam Montembeault
180         Zach Sawchenko
188            Matt Murray
193          Erik Kallgren
194           Joey Daccord
205             J-F Berube
209         Stuart Skinner
211    Mackenzie Blackwood
Name: name, dtype: object>
```

Lopputulemana meillä oli laadukas data, joka piti sisällään 156:n maalivahdin tilasto- ja palkkatiedot.

3.4 Datan kuvaileminen

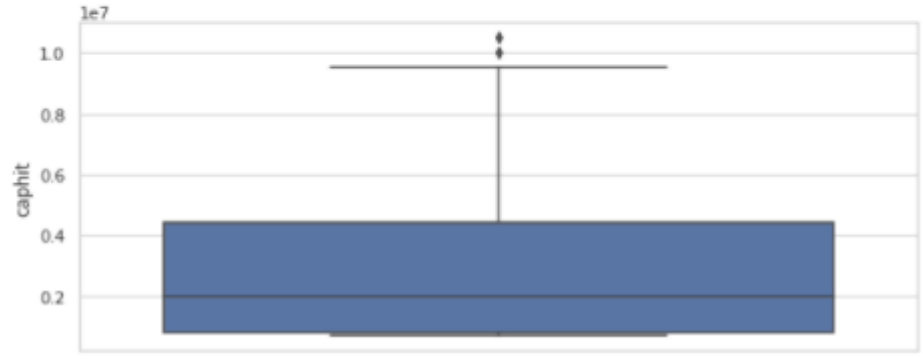
Datan kuvailussa käytettiin vain kauden 21-22 dataa. Tutkittiin miten palkat ovat jakautuneet. Ensin boxplot-kuvaajalla:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()
sns.set_style("whitegrid")
```

```
plt.rcParams["figure.figsize"] = (10,4)
sns.boxplot(data = df21, y = 'caphit')
```

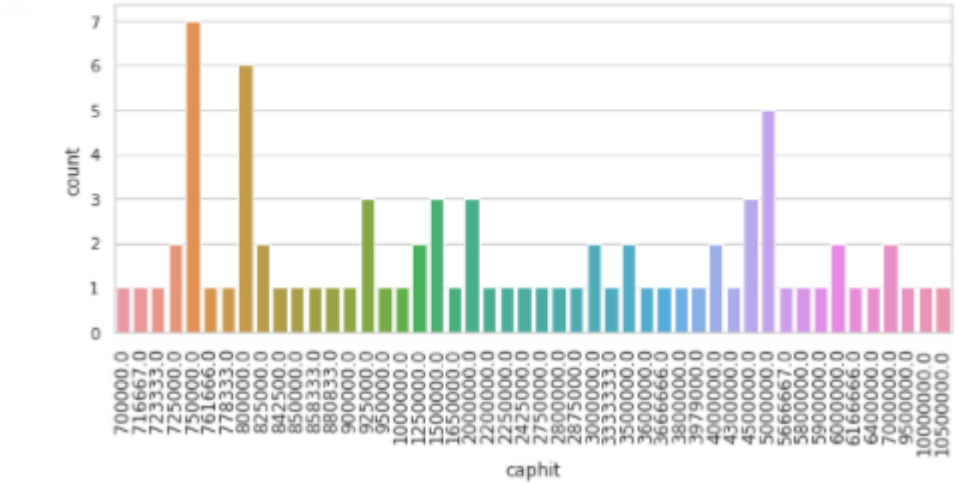
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9716203810>
```



Tämä kuvaaja kertoo hyvin mitkä on minimi ja maksimi palkat, mihin asettuvat suurin osa palkoista ja onko palkoissa selviä *outlier:*itä. 10 miljoonan palkoilla pelaa muutama maalivahti.

Seuraavaksi kuvattiin mitä eri palkkoja maalivahdeilla on:

```
# For this visualisation, sort data by salary:
df21 = df21.sort_values(by=['caphit'], ignore_index=False)
sns.countplot(x = 'caphit', data = df21)
plt.rcParams["figure.figsize"] = (10,3)
plt.xticks(rotation=90, fontsize=12)
plt.show()
```



Tästä huomattiin heti, että 750,000$ on yleinen palkka. Tämä johtuu siitä, että nuo maalivahdit pelaavat ns. tulokassopimuksella, jossa palkka on liigan toimesta määrätty. Koska näiden osalta palkkaus perustuu tulokassopimukseen, eikä varsinaisesti maalivahdin suoritukseen, niin nämä poistettiin tarkkailtavasta datasta:

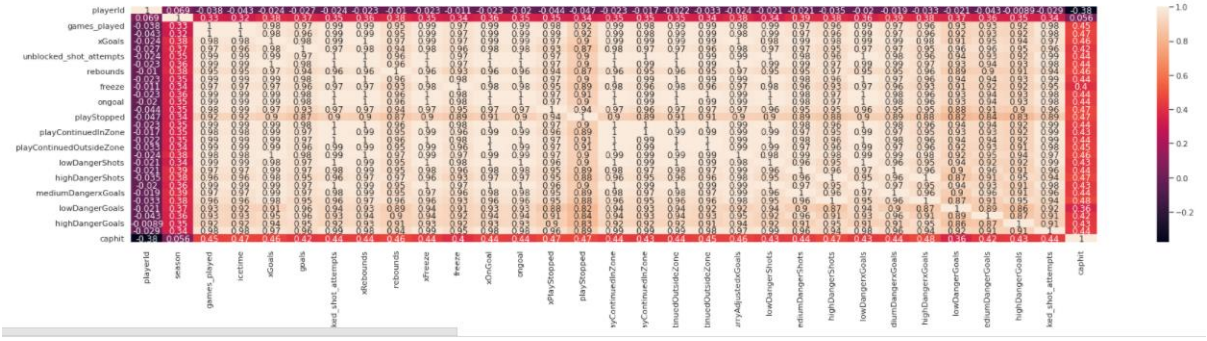
```
148] # make a copy of the dataframe to filter newbies
     df_veterans =df.loc[df['caphit'] != 750000]
     #df_pg_old.drop(df_pg_old[df_pg_old['caphit'] == 750000].index, inplace = True)
     #rslt_df = dataframe.loc[dataframe['Percentage'] != 95]
```

Nyt olemme valmiit mallintamaan dataa.

3.5 Koneoppiminen

Aluksi katsottiin miten datan eri piirteet korreloivat keskenään (alin rivi on palkka (*caphit)*):

```
corrMatrix = df.corr()
sns.set(rc = {'figure.figsize':(30,6)})
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



Alustava havainto ei ollut kovin rohkaiseva. Matriisista ei oikein löytynyt hyviä piirteitä, jotka korreloisivat palkan kanssa. Matriisin lisätutkimus paljasti, että lähes kaikki piirteet korreloivat hyvin pelimäärän (*games_played*) kanssa.

Tehtiin uudet sarakkeet datalle, joihin laskettiin tärkeimpien piirteiden “per peli” arvo. Tärkeimmiksi piirteiksi valittiin:

- päästetyt maalit (per peli)
- rebound-maalit (per peli)
- matalan vaarallisuuden laukauksen maaliolettama (per peli)
- keskinkertaisen vaarallisuuden laukauksen maaliolettama (per peli)
- korkean vaarallisuuden laukauksen maaliolettama (per peli)

```
[153] df_pg['goals_pg'] = df.goals / df.games_played
      df_pg['rebounds_pg'] = df.rebounds / df.games_played
      df_pg['lowDangerxGoals_pg'] = df.lowDangerxGoals / df.games_played
      df_pg['mediumDangerxGoals_pg'] = df.mediumDangerxGoals / df.games_played
      df_pg['highDangerxGoals_pg'] = df.highDangerxGoals / df.games_played
      #df_pg = df_pg.drop(['goals', 'rebounds',  'lowDangerxGoals', 'mediumDangerxGoals', 'highDangerxGoals'], axis=1)
```

```
display(df_pg)
```

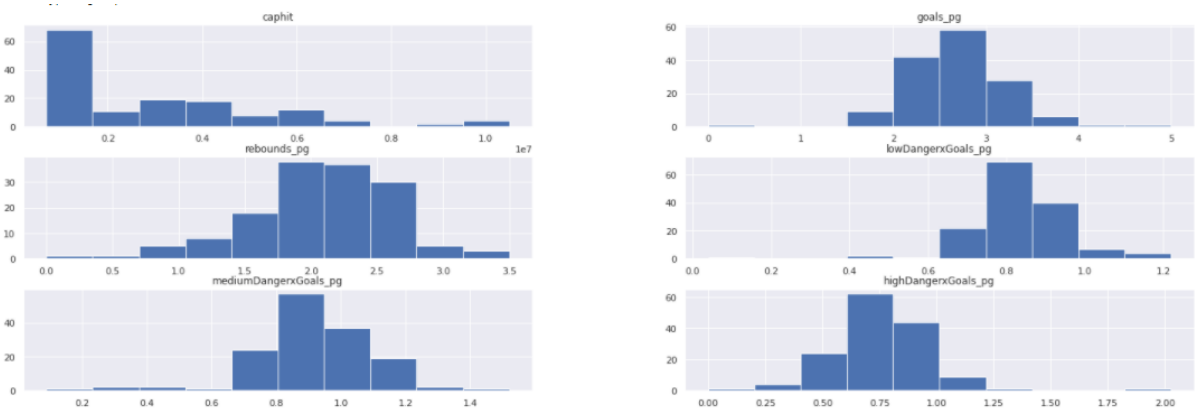
	team	caphit	goals_pg	rebounds_pg	lowDangerxGoals_pg	mediumDangerxGoals_pg	highDangerxGoals_pg
0	T.B	1300000.0	3.083333	1.250000	0.670000	0.926667	0.610833
1	CAR	3400000.0	2.681818	2.000000	0.865455	0.975909	0.550455
2	NSH	1500000.0	2.166667	2.277778	0.888611	0.869722	0.773611
3	WPG	6166666.0	2.488889	2.488889	0.854222	1.174667	0.888000
6	TOR	2750000.0	2.421053	1.789474	0.720000	0.945789	0.593684
...	...	...	...	...	...	...	...

Tarkasteltiin pistematriisilla (*sns.pairplot*) tilanne:

```
sns.pairplot(df_pg, hue="team")
```

```
<seaborn.axisgrid.PairGrid at 0x7f971c172290>
```

Tulos ei näyttänyt koviin lupaavalta. Seuraavaksi tarkasteltiin uusien piirteiden jakaumia:



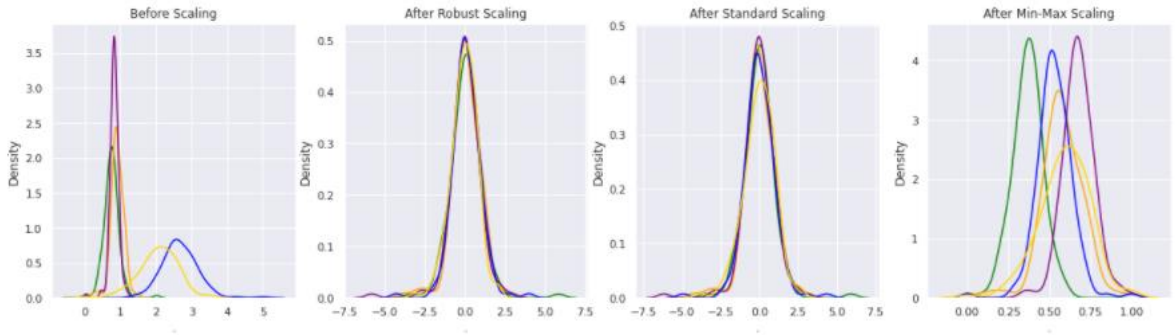
Aloitettiin koneoppiminen jakamalla data labeliin ja piirteisiin:

```
[159] #set the target
      y = df_pg['caphit']
```

```
[160] # set the data
      x = df_pg.drop(['team', 'caphit'], axis=1)
      x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146 entries, 0 to 216
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   goals_pg              146 non-null    float64
 1   rebounds_pg           146 non-null    float64
 2   lowDangerxGoals_pg    146 non-null    float64
 3   mediumDangerxGoals_pg 146 non-null    float64
 4   highDangerxGoals_pg   146 non-null    float64
dtypes: float64(5)
memory usage: 6.8 KB
```

Seuraavaksi ajattelin, että piirteitä skaalaamalla koneoppimismallista tulisi parempi. Halusin kokeilla useita eri tapoja ja visualisoida niiden vaikutukset:



Valitsin *robust*-skaalaajan.

```
# separate df to target and input:
x = robust_df
# y is already defined above
```

Data jaettiin harjoitus- ja testidataan:

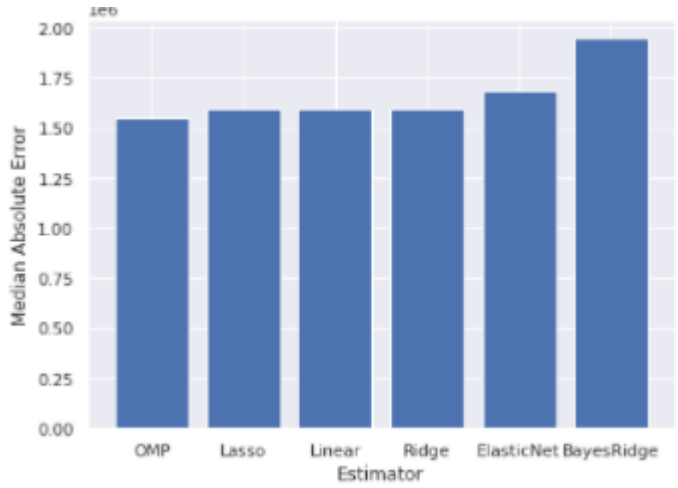
```
# split the data in training and test
from sklearn.model_selection import train_test_split
# splitting the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

Vertailtiin eri regression-menetemiä ja niiden tarkkuutta:

```
from sklearn import ensemble
from sklearn import linear_model
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from collections import Counter
rs = 1
ests = [ linear_model.LinearRegression(), linear_model.Ridge(),
         linear_model.Lasso(), linear_model.ElasticNet(),
         linear_model.BayesianRidge(), linear_model.OrthogonalMatchingPursuit() ]
ests_labels = np.array(['Linear', 'Ridge', 'Lasso', 'ElasticNet', 'BayesRidge', 'OMP'])
errvals = np.array([])

for e in ests:
    e.fit(x_train, y_train)
    this_err = metrics.median_absolute_error(y_test, e.predict(x_test))
    #print "got error %0.2f" % this_err
    errvals = np.append(errvals, this_err)

pos = np.arange(errvals.shape[0])
srt = np.argsort(errvals)
plt.figure(figsize=(7,5))
plt.bar(pos, errvals[srt], align='center')
plt.xticks(pos, ests_labels[srt])
plt.xlabel('Estimator')
plt.ylabel('Median Absolute Error')
```

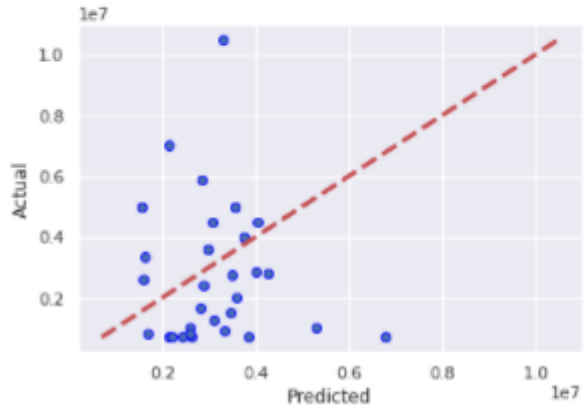


Ylläoleva kuvaaja kertoo, että mikään kokeiltu regressiomenetelmä ei oikein toimi tässä datasetissä. Tämähän oli nähtävissä jo yllä olevissa pistematriiseissa.

Lopuksi varmistettiin asia kuvaajan avulla lineaarista regressiota:

```
y_predicted = model.predict(x_test)
```

```
fig, ax = plt.subplots()
ax.scatter(y_predicted, y_test, edgecolors=(0, 0, 1))
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=3)
ax.set_xlabel('Predicted')
ax.set_ylabel('Actual')
plt.show()
```



Lopputulemana vahvistui epäilys, että maalivahtien palkka NHL:ssä ei perustu pelkästään heidän suorituksiinsa.

Lopuksi malli tallennettiin tiedostoon toimeenpanovaihetta varten:

```
# saving the model
# (https://www.analyticsvidhya.com/blog/2020/12/deploying-machine-learning-models-using-streamlit-an-introductory-guide-to-model-deployment/)
import pickle
pickle_out = open("./lineareg.pkl", mode = "wb")
pickle.dump(model, pickle_out)
pickle_out.close()
```

3.6 Toimeenpano

Toimeenpanoa varten pystytin koneelleni anaconda ja loin uuden ympäristön streamlit:n käyttöä varten.

Seuraavat paketit piti asentaa tähän uuteen ympäristöön:

- streamlit
- pandas
- matplotlib
- plotly

Itse sovelluksen koodin kopioin melko suoraan https://boadziedaniel.medium.com/part-2-the-salary-predictor-a7f4c50a84ca .

Jostain syystä malli ei toiminut ollenkaan streamlit sovelluksessa. Mitä enemmän laittoi maaleja menevän, sitä isompi ehdotettu palkka oli. Aika loppui, enkä päässyt tätä tutkimaan.

4. Lopputulema

Tuli ilmeiseksi, että maalivahdin esitykset eivät välttämättä korreloi palkan kanssa. Vaikka koko harjoituksesta ei jäänyt mitään tuloksen osalta käteen, opin aivan valtavasti. Jälkiviisaana toteuan, että olisi pitänyt ottaa yksinkertaisempi tehtävä.