MINGGU KE 4

Bermain dengan Data

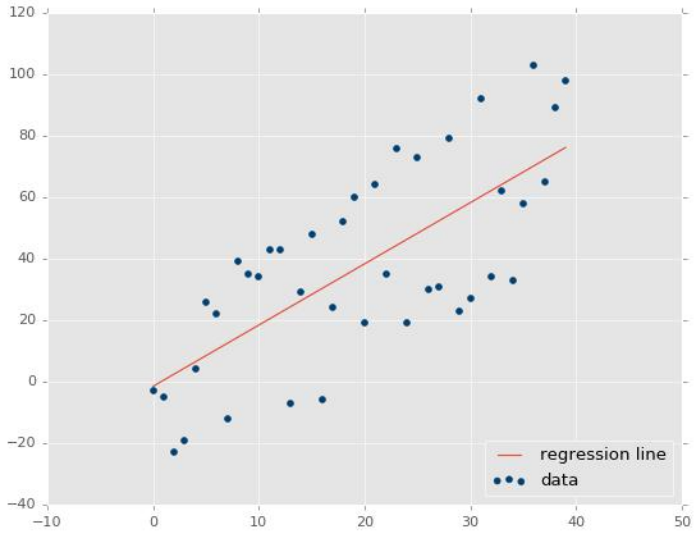
Sebelum memulai pastikan kitas sudah memiliki paket instalasi (pandas), berikut perintah instalasi pandas di Linux via CLI.

```
python3 -m pip install --user pandas
```

Langkah berikutnya kita akan menginstall Quandl. Cara menginstall Quandl adalah sebagai berikut:

```
python3 -m pip install --user quandl
```

Kali ini kita akan mencoba membahas terkait regresi. Tujuannya adalah untuk mendapatkan data kontinu, dan mencari persamaan terbaik dari data tersebut, dan dapat meramal suatu nilai keluaran spesifik. Dengan regresi linear sederhana kita dapat dengan mudah membuat garis yang best fit berikut contohnya:



Dari sini kita akan menggunakan persamaan dari garis tersebut untuk meramalkan dimasa yang akan datang, dimana date adalah x-axis, dan berapa harga yang akan diperoleh.

Regresi linear populer digunakan untuk memprediksi harga saham. Hal ini dilakukan karena kita mempertimbangkan fluiditas dari harga atas dasar waktu, dan mencoba meramalkan aliran harga berikutnya di masa datang menggunakan suatu dataset kontinu.

Regresi adalah suatu bentuk dari supervised learning, yang dimana scientist mengajari mesin dengan menunjukkannya fitur dan kemudian menunjukkannya suatu jawaban yang tepat, dan seterusnya untuk mengajari mesin. Sekali mesin berpikir, scientist biasanya “menguji” mesin pada data

yang tidak terlihat, dimana scientist masih mengetahui jawaban yang tepatnya, tetapi mesin tidak. Jawaban dari mesin akan dibandingkan dengan jawaban yang sudah diketahui, dan akurasi mesin dapat diukur. Jika akurasi cukup tinggi, scientist boleh mempertimbangkan untuk menggunakan algoritme tersebut dalam dunia nyata.

Berikutnya kita akan memulai dengan suatu data sebagai contoh. Kita akan memulai dengan harga saham yang sederhana dan volume informasi dari Quandl. Untuk memulai kita akan menarik data harga saham untuk Alphabet (sebelumnya Google), dengan ticker dari GOOGL, berikut ini contohnya:

```
import pandas as pdimport Quandl
df = quandl.get("WIKI/GOOGL")
print(df.head())
```

Berikut output dari perintah di atas:

Date	Open	High	Low	Close	Volume	Ex-Dividend	\
2004-08-19	100.01	104.06	95.96	100.335	44659000.0	0.0	
2004-08-20	101.01	109.08	100.50	108.310	22834300.0	0.0	
2004-08-23	110.76	113.48	109.05	109.400	18256100.0	0.0	
2004-08-24	111.24	111.60	103.57	104.870	15247300.0	0.0	
2004-08-25	104.76	108.00	103.88	106.000	9188600.0	0.0	

Date	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	\
2004-08-19	1.0	50.159839	52.191109	48.128568	50.322842	
2004-08-20	1.0	50.661387	54.708881	50.405597	54.322689	
2004-08-23	1.0	55.551482	56.915693	54.693835	54.869377	
2004-08-24	1.0	55.792225	55.972783	51.945350	52.597363	
2004-08-25	1.0	52.542193	54.167209	52.100830	53.164113	

Date	Adj. Volume
2004-08-19	44659000.0
2004-08-20	22834300.0
2004-08-23	18256100.0
2004-08-24	15247300.0
2004-08-25	9188600.0

Saat ini kita telah memiliki data yang mungkin tidak banyak, dari data yang kita memiliki kita mempunyai suatu kolom yang sangat kecil dan banyak yang redundan, suatu pasang tidak berubah. Kita akhirnya sepakat bahwa memiliki keduanya antara regular dan adjusted adalah kolom redundan. Kolom adjusted adalah yang paling ideal. Sebetulnya adjusted itu adalah pembagian dua dari

kolom regular.
Berikutnya kita akan memodifikasi data tersebut sebagai berikut:

```
df = df[['Adj. Open',  'Adj. High',  'Adj. Low',  'Adj. Close', 'Adj. Volume']]
```

Hasil dari perintah di atas adalah sebagai berikut :

	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date					
2004-08-19	50.159839	52.191109	48.128568	50.322842	44659000.0
2004-08-20	50.661387	54.708881	50.405597	54.322689	22834300.0
2004-08-23	55.551482	56.915693	54.693835	54.869377	18256100.0
2004-08-24	55.792225	55.972783	51.945350	52.597363	15247300.0
2004-08-25	52.542193	54.167209	52.100830	53.164113	9188600.0

Kita modifikasi dataframe dengan membuat kolom baru yaitu % sebaran berdasarkan closing price.

```
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Low']) / df['Adj. Close']
```

Berikutnya adalah kolom perubahan % harian:

```
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Open'] * 100.0
```

Kemudian kita mendefinisikan suatu dataframe yang baru:

```
df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]print(df.head())
```

	Adj. Close	HL_PCT	PCT_change	Adj. Volume
Date				
2004-08-19	50.322842	0.080730	0.324968	44659000.0
2004-08-20	54.322689	0.079217	7.227007	22834300.0
2004-08-23	54.869377	0.040494	-1.227880	18256100.0
2004-08-24	52.597363	0.076571	-5.726357	15247300.0
2004-08-25	53.164113	0.038868	1.183658	9188600.0

Berikutnya kita akan melakukan regresi pada data harga saham berdasarkan poin yang sebelumnya. Pertama yang akan dilakukan adalah kita akan mengimport modul-modul yang diperlukan seperti berikut ini

```
import quandl, math
import numpy as np
import pandas as pd
from sklearn import preprocessing, cross_validation, svm
from sklearn.linear_model import LinearRegression
```

FITUR DAN LABEL

Kita akan menggunakan modul numpy untuk mengkonversi data ke dalam bentuk array. Kita akan membicarakan masalah praproses (preprocessing) dan

validasi silang (cross_validation) ketika kita ingin mendapatkannya dalam suatu kode, tetapi preprocessing adalah modul yang digunakan untuk melakukan cleaning/scaling dari data utama ke machine learning, dan cross_validation digunakan dalam tahapan pengujian. Kita akan mengimpor algoritme LinearRegression sebaik svm dari Scikit-learn, yang akan digunakan sebagai algoritme pembelajaran mesin untuk mendemokan hasil.

Dalam kasus kita mana yang fitur dan mana yang label? Kita akan mencoba memprediksi harga, maka apakah harga itu adalah label? Jika demikian apa itu fitur? Ketika kita akan meramal harga maka label itu adalah harga di masa datang. Fitur adalah harga saat ini, persen tinggi minus rendah, dan persen perubahan penguapan. Kita akan menambahkan suatu kolom baru sebagai berikut:

```
forecast_col = 'Adj. Close'
df.fillna(value=-99999, inplace=True)
forecast_out = int(math.ceil(0.01 * len(df)))
```

Di sini kita akan mendefinisikan kolom forecasting, kita akan mengisi data NaN dengan -99999. Kita sedikit memiliki pilihan di sini bagaimana menangani data yang missing. Kita tidak dapat melewati saja suatu datapoint NaN (Not a Number) pada suatu pengklasifikasi ML, kita harus menghandle-nya. Satu cara yang populer adalah menggantinya dengan suatu nilai -99,999. Dengan banyak pengklasifikasi ML, hal ini akan dikenali dan diperlakukan sebagai suatu fitur pencilan. Kita dapat juga menghapus semua fitur atau label yang berisi data yang missing, tetapi kita mungkin akan kehilangan banyak dari data keluar.

Dalam dunia nyata, himpunan-himpunan data sangat berantakan. Banyak data saham atau volume adalah bersih, jarang dengan data yang miss, akan tetapi banyak dataset yang memiliki data yang missing.

Akhirnya kita ingin mendefinisikan apa yang akan kita ramalkan keluar. Dibeberapa kasus seperti mencoba untuk memprediksi klien premium asuransi, kita hanya ingin satu angka, untuk “saat ini”, tetapi, dengan meramal, kita ingin meramalkan keluar dengan beberapa angka dari datapoint. Kita akan mengatakan kita ingin meramalkan 1% dari keseluruhan lebar dari dataset. Jika data kita adalah 100 hari dari harga saham, maka kita akan dapat memprediksi harga 1 hari keluar di masa datang. Memilih apasaja yang dinginkan. Jika kita hanya mencoba saja memprediksi harga besok, maka kita akan melakukan 1 hari keluar, dan peramalan akan hanya 1 hari keluar saja. Jika ingin memprediksi 10 hari, kita dapat secara aktual membangkitkan suatu peramalan untuk setiap hari, untuk minggu berikutnya atau setengahnya.

Dalam kasus kita memutuskan fitur adalah kumpulan dari nilai-nilai saat ini, dan label seharusnya harga, dalam masa datang, dimana masa datang adalah 1% dari keseluruhan lebar dataset. Kita akan mengasumsikan semua kolom saat ini adalah fitur kita, lalu kita akan menambahkan suatu kolom baru dengan operasi pandas sederhana berikut ini:

```
df['label'] = df[forecast_col].shift(-forecast_out)
```

Dari perintah di atas kita akan mendapatkan data yang terdiri dari fitur dan label, berikut ini adalah hasilnya:

	Adj. Close	HL_PCT	PCT_change	Adj. Volume	label
Date					
2004-08-19	50.322842	0.080730	0.324968	44659000.0	68.752232
2004-08-20	54.322689	0.079217	7.227007	22834300.0	69.639972
2004-08-23	54.869377	0.040494	-1.227880	18256100.0	69.078238
2004-08-24	52.597363	0.076571	-5.726357	15247300.0	67.839414
2004-08-25	53.164113	0.038868	1.183658	9188600.0	68.912727

Selanjutnya kita akan menghilangkan beberapa informasi NaN dari dataframe:

```
df.dropna(inplace=True)
```

Standard yang digunakan dalam ML bahwa huruf besar sebagai fitur dan huruf kecil sebagai label, sehingga fitur dan label yang akan kita design adalah sebagai berikut:

```
X = np.array(df.drop(['label'], 1))
y = np.array(df['label'])
```

Sebelumnya lebih lanjut kita akan melakukan preprocessing terhadap data sebelum masuk ke training dan testing. Secara umum kita menginginkan nilai fitur kita berada pada rentang -1 sampai 1. Hal ini mungkin tidak akan berarti apa-apa akan tetapi akan lebih mempercepat pemrosesan dan juga dapat membantu akurasi. Karena rentang ini sangat popular digunakan, rentang ini sudah include di dalam modul preprocessing dari Scikit-learn. Untuk mempergunakannya kita dapat mengaplikasinya dengan preprocessing.scale pada variabel X sebagai berikut:

```
X = preprocessing.scale(X)
```

Kemudian membuat label y:

```
y = np.array(df['label'])
```

Sekarang kita akan lanjut ke training dan testing. Cara agar dapat bekerja kita dapat mengambil sebagai contoh 75% dari data dan menggunakannya untuk melatih pengklasifikasi ML. Kemudian kita mengambil sisanya dari data

sebanyak 25% untuk mengujinya. Selama masih dalam contoh data kita seharusnya mendapatkan fitur dan label yang sudah diketahui. Apabila kita menguji 25% data sisanya, kita akan mendapatkan akurasi dan ketahanan yang pendek, atau sering disebut sebagai confidence score (nilai kepercayaan). Ada banyak cara untuk melakukan ini tetapi, kemungkinan cara terbaik adalah dengan menggunakan fungsi build in yang sudah tersedia yaitu cross_validation, sejak ini juga mengajak data untuk kita. Kode untuk melakukan hal ini adalah sebagai berikut:

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
```

Setelah ini kita sudah siap untuk mendefinisikan klasifikasi kita. Banyak pengklasifikasian umum tersedia di Scikit-learn dan bahkan sedikit spesifik untuk regresi. Berikut ini akan ditunjukkan contoh Support Vector Regression dari paket svm Scikit-learn.

```
clf = svm.SVR()
```

Kita akan menggunakan semua default untuk menjaga hal ini sederhana, tetapi kita dapat mempelajarinya lebih lanjut dalam dokumentasi sklearn.svm.SVR. Sekali kita sudah mendefinisikan pengklasifikasi, kita akan siap untuk melakukan training. Dengan sklearn kita akan melatihnya dengan .fit:

```
clf.fit(X_train, y_train)
```

Disini kita akan memfitkan fitur training dan label training. Pengklasifikasi kita sekarang dilatih. Mudah bukan. Sekarang kita akan mengujinya dengan perintah berikut:

```
confidence = clf.score(X_test, y_test)
```

Kemudian kita cetak hasil akurasinya dengan cara berikut:

```
print(confidence) #0.806069791913
```

Disini kita akan mendapatkan nilai akurasi sekitar 80%. Berikutnya kita akan mencoba pengklasifikasi yang lain, kali ini kita menggunakan LineaRegression dari sklearn:

```
clf = LinearRegression()
clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)
print(confidence) #0.974549590614
```

Disini kita mendapatkan akurasi sekitar 97%. Sedikit lebih baik tapi pada dasarnya sama. Kita dapat memodifikasi n_jobs pada LinearRegression seperti berikut ini:

```
clf = LinearRegression(n_jobs=-1)
```

Hal ini memberi arti bahwa algoritme akan menggunakan semua threads yang tersedia. Kemudian kita juga dapat memodifikasi SVR sebagai contoh pada kernel. Berikut ini adalah contohnya

```
for k in ['linear','poly','rbf','sigmoid']:
    clf = svm.SVR(kernel=k)
    clf.fit(X_train, y_train)
    confidence = clf.score(X_test, y_test)
    print(k,confidence)

#outputnya
linear 0.960075071072
poly 0.63712232551
rbf 0.802831714511
sigmoid -0.125347960903
```

Berikutnya kita akan mencoba mekakukan prediksi terhadap data yang belum memiliki target. Beriut adalah contohnya:

```
X = np.array(df.drop(['label'], 1))
X = preprocessing.scale(X)
X_lately = X[-forecast_out:]
X = X[:-forecast_out]

df.dropna(inplace=True)

y = np.array(df['label'])

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
clf = LinearRegression(n_jobs=-1)
clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)print(confidence)
```

Berikutnya kiya akan melakukan prediksi menggunakan perintah sebagai berikut:

```
forecast_set = clf.predict(X_lately)
```

Untuk menampilkan hasilnya dapat kita lakukan dengan cara berikut:

```
print(forecast_set, confidence, forecast_out)
```

```
[ 745.67829395  737.55633261  736.32921413  717.03929303  718.59047951
  731.26376715  737.84381394  751.28161162  756.31775293  756.76751056
  763.20185946  764.52651181  760.91320031  768.0072636   766.67038016
  763.83749414  761.36173409  760.08514166  770.61581391  774.13939706
  768.78733341  775.04458624  771.10782342  765.13955723  773.93369548
  766.05507556  765.4984563   763.59630529  770.0057166   777.60915879]
0.956987938167 30
```

Berikutnya kita akan memplotkan dengan cara menambahkan library berikut ini:

```
import datetime
import matplotlib.pyplot as plt
from matplotlib import style
```

Agar tampilan bagus kita dapat menggunakan style berikut ini:

```
style.use('ggplot')
```

Kita menambahkan kolom baru :

```
df['Forecast'] = np.nan
```

Kita menset nilainya sebagai NaN pada permulaan, tetapi kita akan mempopulasikan beberapa secara singkat. Kita akan memulai meramalkan keesokan harinya (memanggil kembali memprediksi 10% keluaran di masa datang). Kita akan menggrab data di hari terakhir pada dataframe, dan memulai mengassign setiap forcast baru pada suatu hari baru.

```
last_date = df.iloc[-1].name
last_unix = last_date.timestamp()
one_day = 86400    #24×60×60
next_unix = last_unix + one_day
```

Sekarang kita mempunyai hari berikutnya yang kita harap digunakan. Berikutnya kita menambahkan forecast pada dataframe yang ada.

```
for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += 86400
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]
```

Kita sebetulnya melakukan iterasi melalui forecast set, mengambil setiap forecast dan hari dan mengatur nilai-nilainya dalam dataframe (membuat masa datang "features" semua NaN). Kode baris terakhir sederhana saja mengambil semua kolom pertama, mensettingnya ke NaN, dan akhirnya kolom adalah apapun i adalah (ramalan dalam kasus ini). Kita tekah memilih melakukan one-line untuk loop seperti ini sehingga, jika kita memutuskan untuk mengubah dataframe dan fitur, kode masih bisa bekerja. Agar tidak kehilangan buatkan dalam graf.

```
df['Adj. Close'].plot()
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```