Uczenie_maszynowe

July 21, 2020

[36]:
```
#REGRESJA WIELORAKA
```

[1]:
```
import pandas as pd
import numpy as np
#Wszytanie pliku csv przy pomocy biblioteki Pandas
flights2 = pd.read_csv('flight_data_2016_nowe.csv', index_col='New_ID')
#Zmodyfikowanie kolumny UNIQUE_CARRIER na wartości numeryczne pod postacią␣
↪kolumny UNIQUE_CARRIER2
flights2['UNIQUE_CARRIER2'] = flights2['UNIQUE_CARRIER'].astype('category')
flights2.dtypes
flights2['UNIQUE_CARRIER2']=flights2['UNIQUE_CARRIER2'].cat.codes
flights2
```

```
/Users/monikajanocha/opt/anaconda3/lib/python3.7/site-
packages/numpy/lib/arraysetops.py:569: FutureWarning: elementwise comparison
failed; returning scalar instead, but in the future will perform elementwise
comparison
  mask |= (ar1 == a)
```

[1]:

New_ID	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	ARR_DELAY	ORIGIN_CITY_NAME	DEST_CITY_NAME	DISTANCE
0	1	1	3	7	F9	-5.0	Denver, CO	Cedar Rapids/Iowa City, IA	692.0
1	1	1	3	7	F9	19.0	West Palm Beach/Palm Beach, FL	Denver, CO	1679.0
2	1	1	3	7	F9	-2.0	Trenton, NJ	Raleigh/Durham, NC	373.0
3	1	1	3	7	F9	-5.0			
4	1	1	3	7	F9	20.0			
…	…	…	…	…	…	…			
1824398	4	12	30	5	DL	-5.0			
1824399	4	12	30	5	DL	3.0			
1824400	4	12	30	5	DL	-29.0			
1824401	4	12	30	5	DL	-3.0			
1824402	4	12	30	5	DL	-10.0			

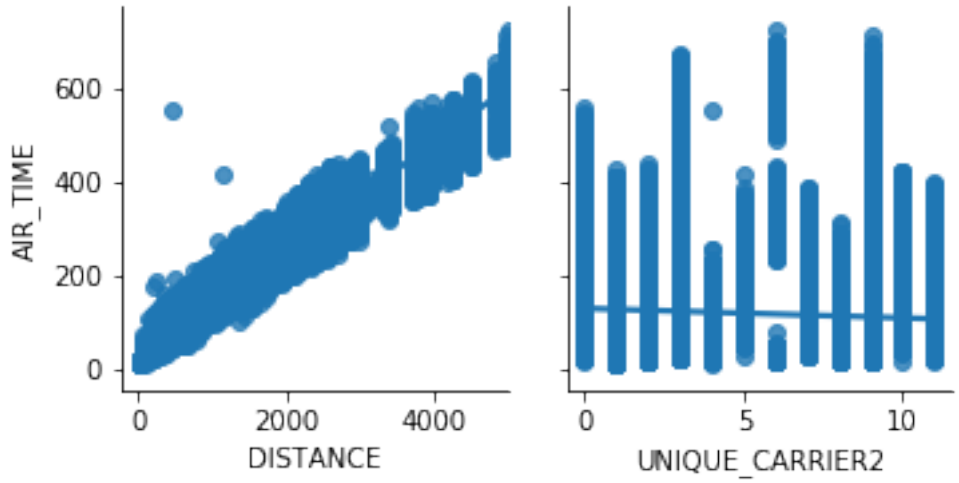
```
3                    Raleigh/Durham, NC                 Trenton, NJ     373.0
4                            Trenton, NJ                Chicago, IL     693.0
…                                    …                        …       …
1824398            Fort Lauderdale, FL                  Atlanta, GA     581.0
1824399                      Atlanta, GA                Milwaukee, WI     669.0
1824400                    Milwaukee, WI                  Atlanta, GA     669.0
1824401                      Atlanta, GA                Fort Myers, FL     515.0
1824402                    Fort Myers, FL                  Atlanta, GA     515.0

         AIR_TIME  air_speed (mph)  UNIQUE_CARRIER2
New_ID
0            87.0       477.241379                5
1           224.0       449.732143                5
2            60.0       373.000000                5
3            57.0       392.631579                5
4           107.0       388.598131                5
…             …                …              …
1824398      88.0       396.136364                3
1824399     105.0       382.285714                3
1824400      89.0       451.011236                3
1824401      71.0       435.211268                3
1824402      85.0       363.529412                3

[1824403 rows x 12 columns]
```

[38]:
```
#Mamy punkt przecięcia i współczynniki. Możemy użyć tych informacji do␣
 ↪zbudowania równania regresji liniowej
#w następujący sposób:
#y = 5.188171624010134 + 0.09926385(DAY_OF_WEEK) -0.07083384(UNIQUE_CARRIER2) -␣
 ↪-0.0011296(DISTANCE)
#Mając nasze równanie/model możemy go użyć do predykcji pensji z nowymi danymi,␣
 ↪np:
```

[2]:
```
import seaborn as sns
sns.pairplot(flights2, x_vars=['DISTANCE','UNIQUE_CARRIER2'],
             y_vars=['AIR_TIME'], kind="reg");
```



[4]:
```
data = flights2[['DISTANCE','UNIQUE_CARRIER2','AIR_TIME']]
data.corr()
```

[4]:

	DISTANCE	UNIQUE_CARRIER2	AIR_TIME
DISTANCE	1.000000	-0.105238	0.982703
UNIQUE_CARRIER2	-0.105238	1.000000	-0.112930
AIR_TIME	0.982703	-0.112930	1.000000

[10]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = data[['DISTANCE','UNIQUE_CARRIER2']]
y = data['AIR_TIME']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)

regressor = LinearRegression()
regressor.fit(X_train, y_train)

print('Intercept:', regr.intercept_)
print('Coefficients:', regr.coef_)
```

```
Intercept: 18.313762146639235
Coefficients: [ 0.11677395 -0.17418475]
```

[11]:
```
y_pred = regressor.predict(X_test)

print('Linear Regression R squared: %.4f' % regressor.score(X_test, y_test))
```

```
Linear Regression R squared: 0.9657
```

[12]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

rmse = sqrt(mean_squared_error(y_test, y_pred))
print('RMSE: %.2f'%rmse )
```

```
RMSE: 13.64
```

[43]:
```
flights2[['UNIQUE_CARRIER', 'UNIQUE_CARRIER2']].drop_duplicates()
```

[43]:

New_ID	UNIQUE_CARRIER	UNIQUE_CARRIER2
0	F9	5
3810	HA	6
4548	DL	3
10261	NK	7
19081	EV	4
21730	OO	8
66886	UA	9
104798	VX	10
108441	AA	0
175984	AS	1
189931	B6	2
218588	WN	11

[17]:
```
DISTANCE = 600
UNIQUE_CARRIER = 0
print ('Przewidywany czas lotu linią AA na dystans równy 600 mil to:',
       'RMSE: %.2f'%regressor.predict([[DISTANCE ,UNIQUE_CARRIER]]))
```

```
Przewidywany czas lotu linią AA na dystans równy 600 mil to: RMSE: 88.38
```

[14]:
```
DISTANCE = 600
UNIQUE_CARRIER = 11
print ('Przewidywany czas lotu linią F9 na dystans równy 600 mil to:',
       'RMSE: %.2f'%regressor.predict([[DISTANCE ,UNIQUE_CARRIER]]))
```

```
Przewidywany czas lotu linią F9 na dystans równy 600 mil to: RMSE: 86.45
```

[15]:
```
DISTANCE = 3000
UNIQUE_CARRIER = 0
print ('Przewidywany czas lotu linią AA na dystans równy 600 mil to:',
       'RMSE: %.2f'%regressor.predict([[DISTANCE ,UNIQUE_CARRIER]]))
```

```
Przewidywany czas lotu linią AA na dystans równy 600 mil to: RMSE: 368.64
```

[16]:
```
DISTANCE = 3000
UNIQUE_CARRIER = 11
print ('Przewidywany czas lotu linią F9 na dystans równy 600 mil to:',
       'RMSE: %.2f'%regressor.predict([[DISTANCE ,UNIQUE_CARRIER]]))
```

```
Przewidywany czas lotu linią F9 na dystans równy 600 mil to: RMSE: 366.70
```

[ ]:
```

```