Model1LR

June 12, 2022

[ ]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
#read data
Data = pd.read_csv('C:/Users/Ahmed Atef/Desktop/data.csv')
print('Data = \n',Data.head(10))
```

```
Data =
          X1        Y1        X2        Y2        X3        Y3        X4  \
0  0.860019 -3.096488  0.753413 -2.530522  0.629331 -1.962419  0.361692
1  0.456454 -2.852605  0.565661 -2.188466  0.568863 -1.593943  0.363601
2 -0.462493 -2.031486 -0.125549 -1.528304  0.123850 -1.026633  0.238231
3  0.106369 -2.186939  0.116188 -1.520864  0.142727 -0.703652  0.068778
4  0.381094 -3.316631  0.390072 -2.754067  0.427469 -2.131407  0.336888
5 -0.262968 -3.256483 -0.094474 -2.594541  0.073604 -1.954448  0.088474
6 -0.970940 -2.595038 -0.683583 -2.042165 -0.376025 -1.472786 -0.190104
7 -0.304093 -1.997585 -0.215963 -1.307632 -0.116897 -0.454085 -0.069022
8  0.055419 -3.592592  0.131205 -2.973669  0.252867 -2.286043  0.242098
9 -0.344932 -3.766955 -0.137432 -2.997226  0.053303 -2.269871  0.066792

         Y4      Angle        X5        Y5        X6        Y6        X7  \
0 -1.227911  15.958005  0.456454 -2.852605  0.565661 -2.188466  0.568863
1 -0.907820  16.424084 -0.462493 -2.031486 -0.125549 -1.528304  0.123850
2 -0.510583  15.635178 -0.915264 -0.867298 -0.556912 -0.619065 -0.347119
3  0.170585  26.299685  0.381094 -3.316631  0.390072 -2.754067  0.427469
4 -1.266161  15.050765 -0.262968 -3.256483 -0.094474 -2.594541  0.073604
5 -1.111491  13.807231 -0.970940 -2.595038 -0.683583 -2.042165 -0.376025
6 -0.800848  14.902200 -1.493574 -1.457450 -1.151122 -1.159289 -0.929195
7  0.416620  15.051214  0.055419 -3.592592  0.131205 -2.973669  0.252867
8 -1.329521  12.915006 -0.344932 -3.766955 -0.137432 -2.997226  0.053303
9 -1.308215   7.824200 -0.905729 -3.185208 -0.604510 -2.520129 -0.291183

         Y7        X8        Y8  Angle number  RightorLeft
0 -1.593943  0.363601 -0.907820             2            0
1 -1.026633  0.238231 -0.510583             3            0
2 -0.385286 -0.006106 -0.007940             4            0
```

```
3 -2.131407  0.336888 -1.266161             1            0
4 -1.954448  0.088474 -1.111491             2            0
5 -1.472786 -0.190104 -0.800848             3            0
6 -0.890322 -0.549895 -0.352399             4            0
7 -2.286043  0.242098 -1.329521             1            0
8 -2.269871  0.066792 -1.308215             2            0
9 -1.861470 -0.161199 -1.063399             3            0
```

[ ]:
```
#show data details
print('Data.describe = \n',Data.describe())
```

```
Data.describe =
                X1           Y1           X2           Y2           X3  \
count  4382.000000  4382.000000  4382.000000  4382.000000  4382.000000
mean     -0.252894    -2.350716    -0.187523    -1.889514    -0.088361
std       0.948101     0.848965     0.822127     0.751828     0.653238
min      -2.702862    -4.136758    -2.085325    -3.225857    -1.476798
25%      -0.831879    -2.977523    -0.673609    -2.433257    -0.496952
50%      -0.146311    -2.435614    -0.075717    -1.984988     0.004254
75%       0.420229    -1.936213     0.418198    -1.596749     0.424851
max       2.221333     0.793691     1.774284     0.824333     1.342360

                Y3           X4           Y4        Angle           X5  \
count  4382.000000  4382.000000  4382.000000  4382.000000  4382.000000
mean     -1.375213    -0.047605    -0.671134    16.798604     0.055876
std       0.696510     0.392837     0.572180     9.504037     0.965893
min      -2.409056    -0.941107    -1.434023     2.763818    -2.157534
25%      -1.882652    -0.400934    -1.053675    10.121567    -0.711105
50%      -1.562080     0.000243    -0.886340    14.579348     0.023831
75%      -1.056059     0.291479    -0.433275    20.187453     0.762173
max       0.771254     0.805917     0.790989    61.767698     2.469773

                Y5           X6           Y6           X7           Y7  \
count  4382.000000  4382.000000  4382.000000  4382.000000  4382.000000
mean     -2.276064     0.114580    -1.893112     0.157911    -1.477540
std       0.825647     0.804275     0.661706     0.665437     0.508988
min      -4.128808    -1.736918    -3.225857    -1.427917    -2.409056
25%      -2.947262    -0.536102    -2.412434    -0.373223    -1.866497
50%      -2.297369     0.093548    -1.953126     0.161692    -1.548207
75%      -1.663655     0.698498    -1.451524     0.622315    -1.156084
max       0.532019     2.074761     0.591465     1.746115     0.604385

                X8           Y8  Angle number  RightorLeft
count  4382.000000  4382.000000   4382.000000  4382.000000
mean      0.101443    -0.804559      2.518028     0.500228
std       0.440341     0.348770      1.106318     0.500057
min      -0.941107    -1.434023      1.000000     0.000000
25%      -0.228869    -1.044162      2.000000     0.000000
```

```
50%       0.107943    -0.879848       3.000000     1.000000
75%       0.408215    -0.577973       3.750000     1.000000
max       1.148080     0.564301       4.000000     1.000000
```

[ ]:
```
# Quick look at the data structure
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4382 entries, 0 to 4381
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   X1             4382 non-null   float64
 1   Y1             4382 non-null   float64
 2   X2             4382 non-null   float64
 3   Y2             4382 non-null   float64
 4   X3             4382 non-null   float64
 5   Y3             4382 non-null   float64
 6   X4             4382 non-null   float64
 7   Y4             4382 non-null   float64
 8   Angle          4382 non-null   float64
 9   X5             4382 non-null   float64
 10  Y5             4382 non-null   float64
 11  X6             4382 non-null   float64
 12  Y6             4382 non-null   float64
 13  X7             4382 non-null   float64
 14  Y7             4382 non-null   float64
 15  X8             4382 non-null   float64
 16  Y8             4382 non-null   float64
 17  Angle number   4382 non-null   int64
 18  RightorLeft    4382 non-null   int64
dtypes: float64(17), int64(2)
memory usage: 650.6 KB
```

[ ]:
```
Data['RightorLeft'].value_counts()
```

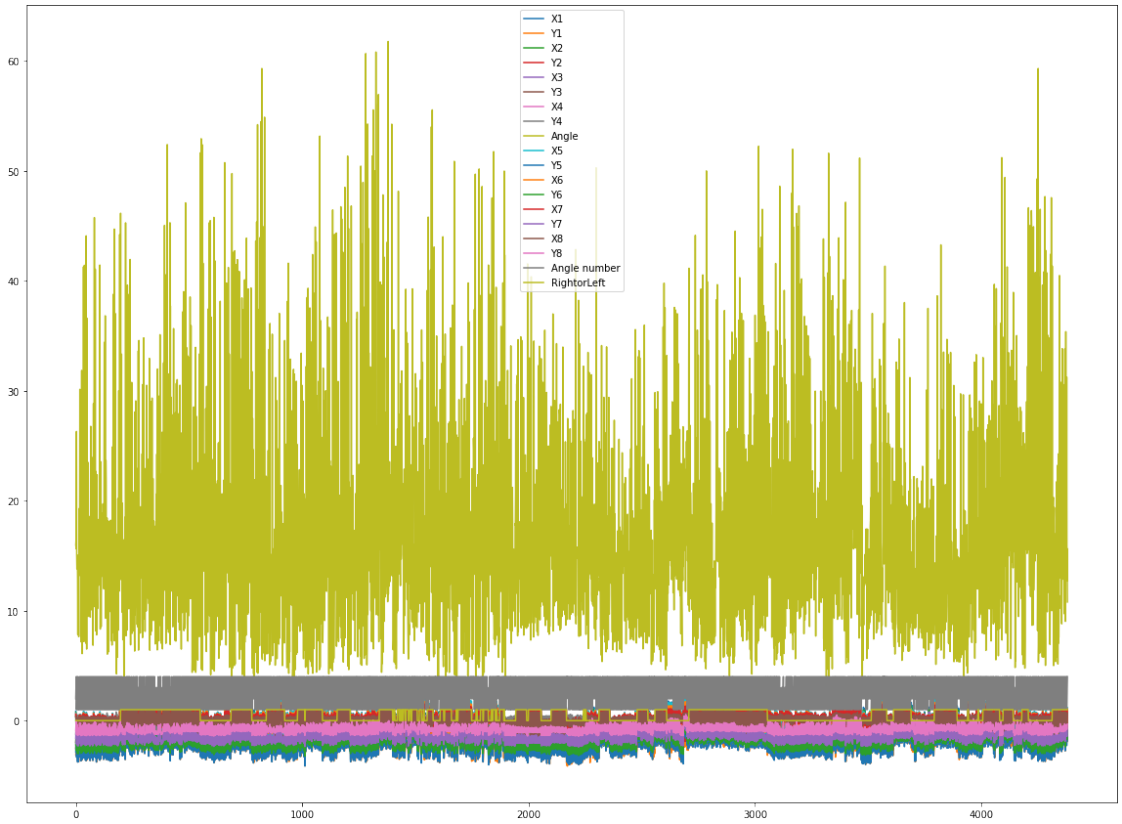
[ ]:
```
1    2192
0    2190
Name: RightorLeft, dtype: int64
```

[ ]:
```
Data['Angle number'].value_counts()
```

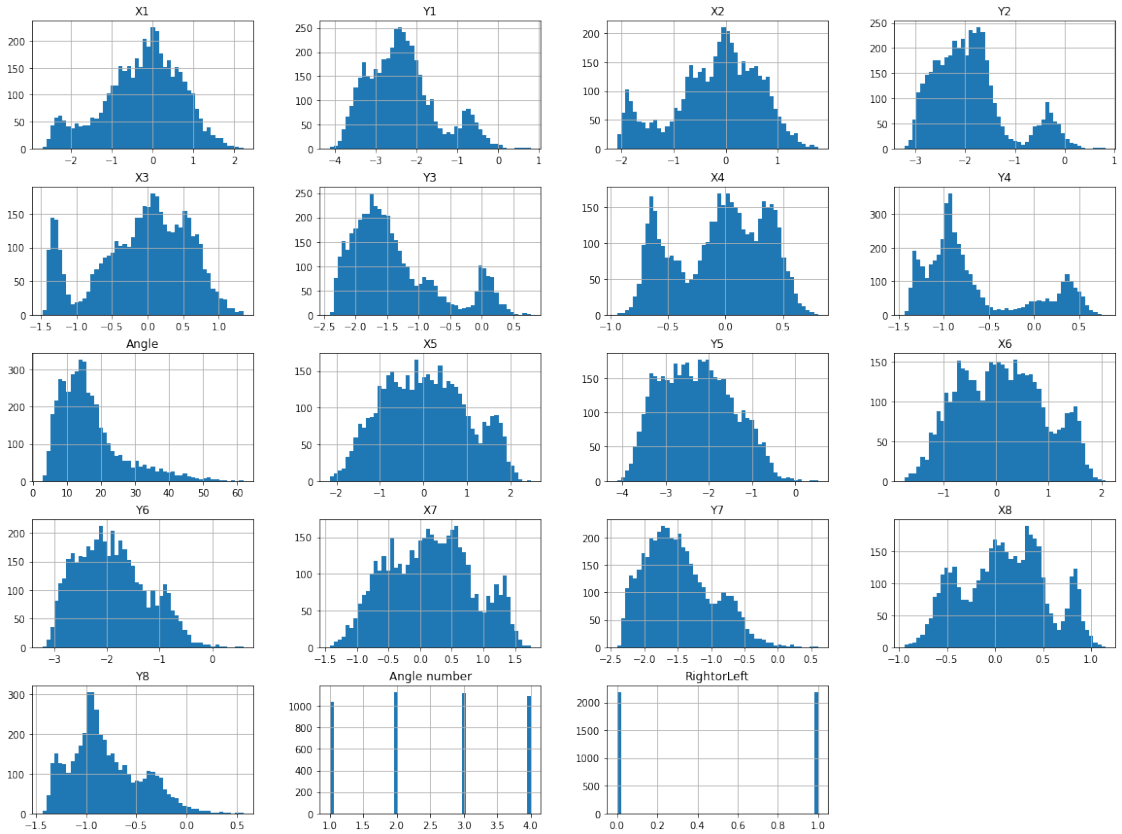
[ ]:
```
2    1132
3    1116
4    1096
1    1038
Name: Angle number, dtype: int64
```

[ ]:
```
Data.plot(figsize=(20,15))
```

[ ]: <AxesSubplot:>



[ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
Data.hist(bins=50, figsize=(20,15))
plt.show()
```



[ ]:

[ ]:
```
# split the data
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(Data, test_size=0.2, random_state=42)
```

[ ]:
```
test_set.head()
```

[ ]:
```
            X1        Y1        X2        Y2        X3        Y3        X4  \
670  -0.036958 -2.439708 -0.151748 -1.751869 -0.203871 -0.937441 -0.286370
2417 -0.496443 -2.212595 -0.277505 -1.715926 -0.049095 -1.199572  0.013035
596  -0.284349 -3.169149 -0.103853 -2.481122  0.073029 -1.829867  0.053820
2629  1.558989 -2.525897  1.278059 -2.063646  0.987350 -1.607824  0.559566
1395  0.921632 -1.839371  0.870753 -1.643973  0.769634 -1.350292  0.511160

            Y4      Angle        X5        Y5        X6        Y6        X7  \
670  -0.047744  39.606503  1.560510 -3.120231  1.263452 -2.564677  0.977038
2417 -0.613426  15.629651 -1.010371 -1.038258 -0.697754 -0.783644 -0.524011
596  -1.061236   4.180429 -0.885141 -2.500813 -0.577390 -1.839822 -0.304928
2629 -1.053691  25.326243  1.193229 -2.051963  1.046683 -1.562685  0.855178
```

```
1395 -0.733900  13.204648  1.370833 -0.738958  1.302034 -0.746930  1.207433

            Y7        X8        Y8  Angle number  RightorLeft
670  -1.978325  0.514782 -1.211725             1            0
2417 -0.556461 -0.271823 -0.118667             4            0
596  -1.263751 -0.188361 -0.663147             3            0
2629 -1.099170  0.454378 -0.651524             2            0
1395 -0.688365  0.869758 -0.309678             4            1
```

[ ]:
```
Data = train_set.copy()
```

[ ]:
```
len(test_set)
```

[ ]: 877

[ ]:
```
len(train_set)
```

[ ]: 3505

[ ]:
```
corr_matrix = Data.corr()
```

[ ]:
```
corr_matrix["Angle"].sort_values(ascending=False)
```

[ ]:
```
Angle           1.000000
Y4              0.673085
Y3              0.633347
Y2              0.551183
Y1              0.448352
RightorLeft    -0.007250
X5             -0.046684
X6             -0.083856
Y5             -0.103842
X7             -0.117606
Y6             -0.129496
Y7             -0.149414
X8             -0.171393
Y8             -0.207860
X4             -0.324361
X1             -0.355447
X3             -0.373661
X2             -0.378034
Angle number   -0.491408
Name: Angle, dtype: float64
```

[ ]:
```
# Prepare the data for Machine Learning algorithms

Data = train_set.drop("Angle", axis=1) # drop labels for training set
Data_label = train_set["Angle"].copy()
```

[ ]:
```
Data_label
```

[ ]:
```
647      8.964206
2194    18.802308
2244    18.284985
4256    15.296240
1164    16.180668
           …
3444    15.691533
466      6.504340
3092    41.578601
3772    14.907241
860      5.382245
Name: Angle, Length: 3505, dtype: float64
```

[ ]:
```
# Select and train a model

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(Data, Data_label)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
some_data = Data.iloc[:5]
some_labels = Data_label.iloc[:5]
print("Predictions:", lin_reg.predict(some_data))
```

```
Predictions: [ 9.51326826 11.45098353 11.91521907 14.17160773 13.56828679]
```

[ ]:
```
# Compare against the actual values:

print("Labels:", list(some_labels))
```

```
Labels: [8.964205768, 18.80230809, 18.28498482, 15.29624034, 16.1806679]
```

[ ]:
```
from sklearn.metrics import mean_squared_error

Data_predictions = lin_reg.predict(Data)
lin_mse = mean_squared_error(Data_label, Data_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

[ ]:
```
5.675163465130474
```

[ ]:
```
# Cross Validation

from sklearn.model_selection import cross_val_score
```

```
lin_scores = cross_val_score(lin_reg, Data, Data_label,
                             scoring="neg_mean_squared_error", cv=10)
lin_rmse_scores = np.sqrt(-lin_scores)
```

[ ]:
```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

display_scores(lin_rmse_scores)
```

```
Scores: [5.5514259  5.70120708 5.636896   5.81338988 5.43271082 5.9824288
 5.82029774 5.59668624 5.9919363  5.58201444]
Mean: 5.710899320110563
Standard deviation: 0.17698286810221756
```

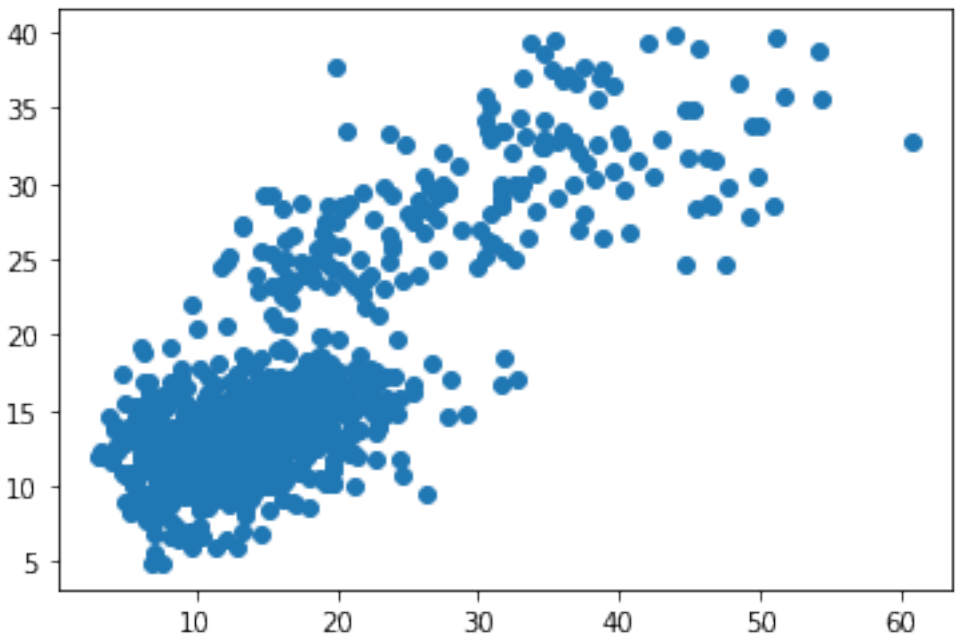
[ ]:
```
X_test = test_set.drop("Angle", axis=1)
y_test = test_set["Angle"].copy()

final_predictions = lin_reg.predict(X_test)

final_mse = mean_squared_error(y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
```

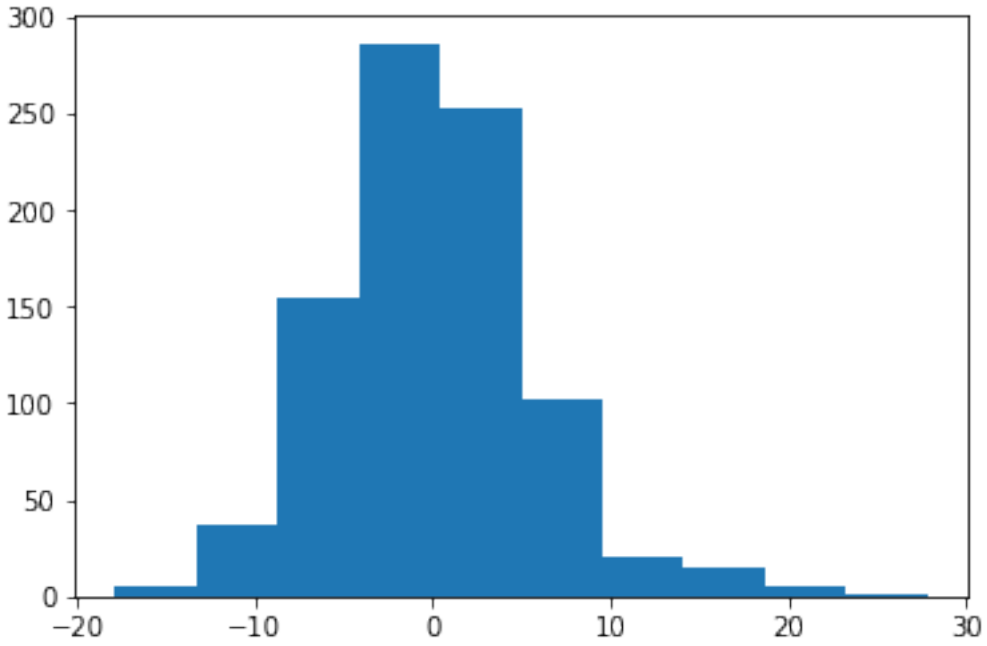
[ ]:
```
plt.scatter(y_test, final_predictions)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x1ce4413ba30>
```



[ ]:
```
plt.hist(y_test - final_predictions)
```

[ ]:
```
(array([  5.,  37., 154., 286., 252., 102.,  20.,  15.,   5.,   1.]),
 array([-17.84574147, -13.27659286,  -8.70744425,  -4.13829565,
          0.43085296,   5.00000157,   9.56915017,  14.13829878,
         18.70744739,  23.27659599,  27.8457446 ]),
 <BarContainer object of 10 artists>)
```



[ ]:
```
final_rmse
```

[ ]:
```
5.806596073732241
```

[ ]:
```
from scipy import stats

confidence = 0.95
squared_errors = (final_predictions - y_test) ** 2
np.sqrt(stats.t.interval(confidence, len(squared_errors) - 1,
                         loc=squared_errors.mean(),
                         scale=stats.sem(squared_errors)))
```

[ ]:
```
array([5.43405513, 6.15663551])
```

[ ]:
```
import sklearn.metrics as sm
```

```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test,␣
 ↪final_predictions), 2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test,␣
 ↪final_predictions), 2))
mean_test=round(sm.mean_squared_error(y_test, final_predictions), 2)
print(np.sqrt(mean_test))
print("Median absolute error =", round(sm.median_absolute_error(y_test,␣
 ↪final_predictions), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,␣
 ↪final_predictions), 2))
print("R2 score =", round(sm.r2_score(y_test, final_predictions), 2))
```

```
Regressor model performance:
Mean absolute error(MAE) = 4.4
Mean squared error(MSE) = 33.72
5.80689245638319
Median absolute error = 3.49
Explain variance score = 0.62
R2 score = 0.62
```

[ ]:
```
def mean_absolute_percentage_error(y_test, final_predictions):
    y_test, final_predictions = np.array(y_test), np.array(final_predictions)
    return np.mean(np.abs((y_test - final_predictions) / final_predictions)) *␣
 ↪100
```

[ ]:
```
print(mean_absolute_percentage_error(y_test, final_predictions))
```

[ ]:
```

```