Hilal Işık

Trial Lecture

PYTHON LINEAR REGRESSION MODEL CHEAT SHEET

WHICH LIBRARIES TO IMPORT

```
# DATA LIBRARIES
import pandas as pd
import numpy as np
# VISUALISATION LIBRARIES
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# MODELLING LIBRARIES
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae
```

PRELIMINARY OPERATIONS

```
df=pd.read_excel('data.xls')  #read xls file
df=pd.read_csv('data.csv')    #read csv file

#DISPLAY THE FIRST 5 ROWS
df.head()

#CHEKING HOW MANY ROWS AND COLUMNS DF HAS
df.shape

#GETTING DF INFO
df.info()
```

TRAIN THE MODEL

```
# CREATE X and y
y=df.col1                    #create df var to predict
X=df.drop('col1', axis=1)  #create df features
# SPLIT DATASET INTO TRAIN AND TEST
X_train, X_test, y_train, y_test =train_test_split(X, y,test_size=0.3,
random_state=123)
# STANDARDIZE THE DATA IF NEEDED
scaler=StandardScaler().fit(X_train)
# we want to apply the exact same scaling as for your training data
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)
# FIT THE MODEL
model = LinearRegression()
model.fit (X_ train, y_train) #train/fit the model
# SHOW THE RESULT
model.intercept_
model.coef_
# PREDICTIONS
y_pred_train=model.predict(X_train)
y_pred=model.predict(X_test)
```

NOTES

 # FITTING THE MODEL: we fit our model only to training set NOT to the test set

 # STANDARDIZATION: we want to apply the exact same scaling as for your training data, therefore we fit the scaler to the training data and use the same object to scale train and test

 # UNDERFITTING

high training and high testing error

 # OVERFITTING

extremely low training error but a high testing error

EVOLUTION METRICS

```
Train_R2=r2_score(y_train,y_pred_train)
Test_R2=r2_score(y_test,y_pred)
MAE = mae(y_train,y_pred_train) #mean absolute error
MSE = mse(y_train,y_pred_train) #mean squared error
```

VISUALISATION

```
#DISTRIBUTION OF THE VARIABLE
sns.displot(df, x="col1")

#SUBPLOTS
fig, ax = plt.subplots(2,2, figsize = (10,13))
sns.histplot(x=df['verb_SAT'], kde = True, ax=ax[0,0])
sns.histplot(x=df['high_GPA'], kde = True, ax=ax[0,1])
sns.histplot(x=df['comp_GPA'], kde = True, ax=ax[1,0])
sns.histplot(x=df['math_SAT'], kde = True, ax=ax[1,1])
```

```
#CORRELATION MATRIX
corr = df.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(9, 7))
    ax = sns.heatmap(corr, mask=mask,cmap='coolwarm',
vmin=-1,vmax=1,annot=True, square=True)

#SEABORN RESIDUAL PLOT
plt.figure(figsize=(10,5))
sns.regplot(x='y_pred',y='y_test', data=result,
scatter_kws={"color": "#C8A2C8"}, line_kws=
{"color":'purple'})
```