Linear Modeling Flow Process + Python Code

@dupadhyaya

LM: Small Steps

- Data (IV + DV)
- Predict DV ~ IVs

Linear Regression Parts

- Read/ Clean Data
- Load Libraries
- Create Model
- Predict Values
- Accuracy/ Predictability - RMSE, R2

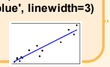
LM : Data Partition

- Data (IV + DV)
- Predict DV ~ IVs
- 2 Sets of Data
  - Train(75%)
    - IVs: train_X
    - DVs: train_y
    - Model (DV ~ IVs)
    - Verify Model Output (F-Statistics, p-Values, AIC, R2, Adjusted R2
    - Check for Assumptions
  - Test (25%)
    - IVs: test_X
    - DVs: test_y
- Predict DV based on test_X
- Predicted_y (on test_X)
- How Accurate is our Model
- Calculate Accuracy in Prediction (RMSE & Other Scores)
- Deploy the Model (if Accuracy is acceptable) : Use it for known Xs and unknown Ys

from sklearn import linear_model

#Library1: sklearn::linear_model

```
model1 = linear_model.LinearRegression()
model1.fit(X,y)                      # train Model
y_pred1 = model1.predict(X)          # predict y on existing Xs
model1.coef_                         # Coefficients: [[1.669]]  #b1 coef
model1.intercept_                    # array([0.964])  #b0 coef
print("Mean squared error: %.2f" % mean_squared_error(y, y_pred1))   # Mean squared error: 0.80
print('Variance score: %.2f ' % r2_score(y, y_pred1))                # Variance score: 0.90
plt.scatter(X, y,  color='black')
plt.plot(X, y_pred1, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show();
```



Compare Values

Python Code

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.formula.api import ols
import statsmodels.api as sm
from pydataset import data
```

- Libraries
- Data
  - one IV : area-rent
  - Multiple IVs
- Bunch Data

https://raw.githubusercontent.com/DUanalytics/datasets/master/csv/slr1.csv

#Data Preparation

```
data = pd.read_csv( url)
X = data.X.values
X=X.reshape(-1,1)
y = data.Y.values
y=y.reshape(-1,1)
```

import statsmodels.api as sm

#Library2 : with statsmodels

#without constant term

```
model2 = sm.OLS(y, X).fit()
predictions2 = model2.predict(X)
predictions2
model2.summary()
```

#with constant term b0

```
X2 = sm.add_constant(X)
model3 = sm.OLS(y, X2).fit()
model3.summary()
predictions3 = model3.predict(X2)
predictions3
```

from statsmodels.formula.api import ols

#Library3: statsmodels.formula.api

#with Diagnostic Plots & constant term

```
data
model4 = ols('Y ~ X', data=data).fit()
model4.summary()
```

import seaborn as sns

```
sns.set_style('darkgrid')
```

import matplotlib.pyplot as plt

```
fig= plt.figure(figsize=(15,8))
```

#diagnostic plots

```
fig = sm.graphics.plot_regress_exog( model4, 'X', fig=fig)
fig
```

Multiple IVs

#Libraries

```
from sklearn import linear_model as lm
from statsmodels.formula.api import ols
from pydataset import data
from sklearn.metrics import mean_squared_error, r2_score
```

#dataset

```
mtcars = data('mtcars')
df1 = mtcars[['mpg','wt','hp']]
```

#LM with OLS

```
MTmodel1 = ols("mpg ~ wt + hp", data=df1).fit()
print(MTmodel1.summary())
```

#Prediction

```
predictionM1 = MTmodel1.predict()
predictionM1
```

#Plots

- #Fit Plot
  - fig, ax = plt.subplots(figsize=(12, 8))
  - fig = sm.graphics.plot_fit(MTmodel1, "wt", ax=ax)
- Regression Plots
  - fig= plt.figure(figsize=(15,8))
  - fig = sm.graphics.plot_regress_exog(MTmodel1, 'wt', fig=fig)
- Component-Component plus Residual (CCPR) Plots
  - x
    - fig, ax = plt.subplots(figsize=(12, 8))
    - fig = sm.graphics.plot_ccpr(MTmodel1, "wt", ax=ax)
  - x
    - fig = plt.figure(figsize=(12, 8))
    - fig = sm.graphics.plot_ccpr_grid(MTmodel1, fig=fig)

#Partition Data

```
IV = df1[['wt','hp']].values        # IV
DV= df1['mpg'].values               # DV
IV_train, IV_test, DV_train, DV_test = train_test_split(IV, DV,test_size=0.2, random_state=123)
IV_train   IV_test   DV_train   DV_test
```

sklearn: linear_model

#Model with Partition

```
MTmodel2a = linear_model.LinearRegression()
MTmodel2a.fit(IV_train, DV_train)
MTmodel2a.intercept_    MTmodel2a.coef_
```

#Predict

```
predicted2a = MTmodel2a.predict(IV_test)        predicted2a
#The mean squared error
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_squared_error
mean_squared_error(DV_test, predicted2a)
from sklearn.metrics import r2_score
r2_score(DV_test, predicted2a)
```