Install Jupyter

```
go to
C:\Users\Admin\AppData\Local\Programs\Python\Python311\Scripts

-> pip install jupyter

-> jupyter notebook
-> python -m notebook
```

Installing Libraries

```
pip list

pip install numpy
pip install pandas
pip install scikit-learn
pip install tensorflow
pip install seaborn
```

sklearn

```
# General
from sklearn import datasets                                    #
datasets.load_databasename()
from sklearn.datasets import fetch_openml

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report
from sklearn.preprocessing import LabelEncoder

# Linear & Polynomial Regration
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.preprocessing import PolynomialFeatures

from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

# Classification
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

```
# Clustering
from sklearn.cluster import KMeans

# PCA
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
```

1. Linear Regression:

```
lg = LinearRegression()
lg.fit(X_train, Y_train)

print(lg.coef_, lg.intercept_)

Y_predict = lg.predict(X_test)
r2_score(Y_predict, Y_test)
```

2. Polynomial regression:

```
poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X_train)

lr = LinearRegression()
lr.fit(X_poly, Y_train)

X_test = np.linspace(-10, 10, 100).reshape(-1,1)

X_transformed = poly_features.transform(X_test)
Y_predict = lr.predict(X_transformed)
```

3. Logistic Regression:

```
lgr = LogisticRegression()
lgr.fit(X_train, Y_train)

Y_pred = lgr.predict(X_test)

cm = confusion_matrix(Y_test, Y_pred)
ac = accuracy_score(Y_test, Y_pred)
cr = classification_report(Y_test, Y_pred)

FPR = cm[0][1] / (cm[0][1] + cm[0][0])
TPR = cm[1][1] / (cm[1][1] + cm[1][0])
```

4. Svm:

```
clf = svm.SVC()
clf.fit(X_train, Y_train)

Y_predict = clf.predict(x_test)
```

5. Clustering:

```
kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(X_train)

# Visualize the clustering
plt.scatter(X_train[:, 0], X_train[:, 1], c=labels, s=10)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100,
c='red')
plt.show()
```

```
# Elbow Method (For number of clusters)

cwss = []

for i in range(1, 11):
    kms = KMeans(n_clusters=i, random_state=0)
    kms.fit(df)
    cwss.append(kms.inertia_)

plt.plot(range(1,11), cwss)
plt.show()
```

```
kmeans = KMeans(n_clusters=3, random_state=0)
df['Cluster'] = kmeans.fit_predict(df)

sbn.scatterplot(data=df, x='feature1', y='feature1', hue='Cluster')
```

Numpy

```
Numpy Array
    create numpy array  :- array(list),
    dimensions          :- 1D, 2D, 3D

Array creation using built-in function
    np.array(list)
    np.arange(start,stop,step,dtype)
    np.zeros(shape, dtype, order)
    np.ones(shape, dtype, order)
    np.linspace(start,stop,num)         # [start, ..., end], length -> points
    np.eye(n,m, k,dtype)

Attributes of numpy array
    array.ndim
    array.shape
    array.size
    array.dtype
    array.itemsize
    array.T

    np.c_[nparray, nparray]             # concatenate alog the column
    np.r_[nparray, nparray]             # concatenates along the row
```

```
Array Manipulation
    np.resharow-count pe(array, newshape, order)
    np.resime in ze(array, newshape)
    array.flatten(order)
    np.concatenate(arrays, axis)
    np.vstack(arrays)
    np.hstack(arrays)
    np.split(array,num-sub-array)
    np.insert(array,index,value,axis=None)
    np.append(array,value,axis=None)
    np.delete(array,index,axis=None)

    np.mean(nparray)
    np.max(nparray)
    np.min(nparray)

    np.random.rand(rows,cols)       # uniform distribution, random number
between 0 and 1
    np.random.randn(rows,cols)      # standard normal distribution, both
positive and negative numbers,
                                    # (mean 0, standard deviation 1)
    np.random.randint(low, high)    # Return random integers from low
(inclusive) to high (exclusive).

Array Indexing
    array[index]                    1D
    array[inde][index]              2D
    array[index][index][index]      3D

Array Iterating
    for loop
    np.nditer(array,order)
    np.ndenumerate(array,order)
```

Matplot

```
import matplotlib.pyplot as plt
```

```
plt.plot(X, Y, color='#58b970', label='Regression Line')
plt.scatter(X, Y, c='#ef5423', label='Scatter Plot')
plt.imshow(any_image, cmap='binary')
plt.pie(data, explode=None, labels=None, colors=None, autopct=None,
shadow=False)
plt.bar(x-labels,height,width,bottom,align)

plt.xlabel('Head Size in cm3')
plt.ylabel('Brain Weight in grams')

plt.legend()
plt.show()
```

How to change heatmap size:

```
plt.figure(figsize=(14, 12))
sbn.heatmap(df.corr(), annot=True, cmap='coolwarm', )
plt.show()
```

Seaborn

```
import seaborn as sbn
```

```
sbn.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

Pandas

```
Pandas Series:
    pd.Series(list|dict|array|csv)
series


Data Frame:
    pd.read_csv('data.csv')
data frame
    pd.read_excel('data.xlsx')
data frame
    pd.DataFrame(dict|list|array, column=['title1', 'title2'])
data frame
    pd.DataFrame(data=np.c_[X,Y], column=['title1', 'title2'])
data frame

    daraframe.loc[index|key]
row

    to_csv("new.csv")
    to_excel("new.xlsx")

Combining data frames
    pd.concat([df1, df2], axis=0)
data frame
    pd.merge(df1, df2, on='key_column', how='inner|left|right|outer')
data frame
    df.append(df1)
data frame

Manipulating DataFrame:
    dataframe.head(row-count)
    dataframe.describe()
    dataframe.sort_values(by,ascending)
    dataframe.isnull()
    dataframe.dropna()
    dataframe.fillna(Scalar|dict|Series|DataFrame|None)

    dataframe.sum()
    dataframe.mean()
    dataframe.median()

    dataframe.corr()                                    # correlation

    dataframe.shape
```

Handle NULL values:

```
dataframe.isnull().sum()

df['colum_title'] = df['colum_title'].fillna(df['colum_title'].mean())
```

Convert categorical data to Numerical Value

```
# Identifying column with non numeric data

df.select_dtypes(exclude=['number'])
```

```
df['gender'].nunique()
df['gender'].unique()

df['gender'] = df['gender'].map({
    'MALE': 0,
    'FEMALE': 1
})
```

```
# Converting categorical data to numerical data

from sklearn import preprocessing

categorical = ['workclass', 'education', 'marital.status', 'occupation',
'relationship', 'race', 'sex']

for feature in categorical:
    le = preprocessing.LabelEncoder()
    X_train[feature] = le.fit_transform(X_train[feature])
    X_test[feature] = le.transform(X_test[feature])
```

```
# Replace '?' with NaN

import numpy as np
df.replace('?', np.nan, inplace=True)
```

Split data into features (X) and target (y)

```
X = df.drop(columns=["target_column"])
X = df[['col1', 'col2',...]]

Y = df["target_column"]
```