Linechart

```
import matplotlib.pyplot as plt

hours = [10, 9, 2, 15, 10, 16, 11, 16]

score = [95, 80, 10, 50, 45, 98, 38, 93]

plt.plot (hours, score, marker = '*', color = 'red', linestyle = 'dotted')
plt.xlabel ("No. of Hours Studied")
plt.ylabel (" Score in final exam ")
plt.title ("Effect of hours studied on Exam score")

plt.grid(True)
plt.show()
```

- - - - - -

Histogram

```
import pandas as pd
import matplotlib.pyplot as plt

mtcars = pd.read_csv(" dataset/mtcars.csv")
mtcars
mtcars.head()

plt.hist(mtcars['mpg'], color='skyblue', edgecolor='black', bins=10)
plt.xlabel ("Miles per gallon (mpg) ")
plt.ylabel (" Frequency ")
plt.title (" Histogram of Miles per gallon (mpg)")

plt.show()
```

Data Cleansing & Preprocessing

```
import pandas as pd
import numpy as np
import re

1. df = pd.read_csv('dataset/BL-Flickr-Images-Book.csv')

print('Original dataframe:', df.head())

2. columns_to_drop = ['Edition Statement', 'Corporate Author', 'Corporate
                      Contributors', 'Former owner', 'Engraver', 'Contributors',
                      'Issuance type', 'Shelfmarks']

df.drop(columns = columns_to_drop, inplace = True)

print('Dataframe after dropping irrelevant columns:', df.head())

3. df.set_index('Identifier', inplace = True)
print("Dataframe after setting index as 'Identifier':", df.head())

4. df['Date of Publication'] = df['Date of Publication'].str.extract(r'^(\d{4})',
                                                                     expand = False)

print("Dataframe with cleared 'Date of Publication':", df.head())

5. df['Place of Publication'] = np.where(df['Place of Pub'].str.contains('London'), 'London',
                                         np.where(df['Pop'].str.contains('Oxford'), 'Oxford',
                                                  df['Place of Publication']))

print('Final cleaned Dataframe:', df.head())
```

Logistic Regression

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScalar
from sklearn.pipeline import make_pipeline

iris = load_iris()
X = iris.data
y = iris.target

df = pd.DataFrame(data = X, columns = iris.feature_names)
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2,
                                                    random_state = 42)

pipeline = make_pipeline(StandardScalar(), LogisticRegression(C=1e4,
                                                    max_iter = 1000))

pipeline.fit(X_train, y_train)

accuracy = pipeline.score(X_test, y_test)
print("Classification accuracy:", accuracy *100)
```

SVM Classifier

```
import pandas as pd
from sklearn.datasets import load_iris()
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

iris = load_iris()
X = iris.data
y = iris.target
column = iris.feature_names

df = pd.DataFrame(data = X, columns = column)
df.head()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 42)

hyperparameters = [
    {'kernel' : 'rbf', 'gamma' : 0.5, 'C' : 0.01},
    {'kernel' :      — " —          'C' : 1},
                — " —                 10},
]

best_accuracy, best_model, best_support_vectors = 0, None, None

for p in hyperparameters:
    model = SVC(kernel = p['kernel'], gamma = p['gamma'], C = p['C'],
                decision_function_shape = 'ovr')

    model.fit(X_train, y_train)
    accuracy = model.score(X_test, y_test)
    support_vectors = model.n_support_.sum()
    print('For hyperparameters = {params}, Accuracy, support_vectors
    if accuracy > best_accuracy:
        print('Best accuracy & Total support vectors')
```

Decision Tree Classifier based on ID3 algorithm

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from io import StringIO
from IPython.display import Image
import pydotplus

data = pd.read_csv('dataset/dataset.csv')
df = pd.DataFrame(data)
print(df)

df = pd.get_dummies(df, columns=['Price', 'Maintenance', 'Airbag'])

X = df.drop('Profitable', axis=1)
y = df['Profitable']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data, filled=True, rounded=True,
                special_characters=True, feature_names=X.columns)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Clustering

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import adjusted_rand_score

data = np.loadtxt('dataset/spiral.txt', delimiter=',', skiprows=1)
df = pd.DataFrame(data)
df.head()

X = data[:, :2]
y_true = data[:, 2]

plt.figure(figure_size = (8,6))
plt.scatter(X[:,0], X[:,1], c=y_true, cmap='viridis')
plt.title('True Clusters')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_clusters = kmeans.fit_predict(X)

single_link = AgglomerativeClustering(n_clusters=3, linkage='single')
single_clusters = single_link.fit_predict(X)

complete_link = AgglomerativeClustering(n_clusters=3, linkage='complete')
complete_clusters = complete_link.fit_predict(X)

rand_index_kmeans = adjusted_rand_score(y_true, kmeans_clusters)
rand_index_single = adjusted_rand_score(y_true, single_clusters)
rand_index_complete = adjusted_rand_score(y_true, complete_clusters)
print
```