**Multiple linear regression**

import numpy as np

import pandas as pd

dataset = pd.read\_csv('50\_Startups.csv')

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, 4]

states=pd.get\_dummies(X['State'],drop\_first=True)

X=X.drop('State',axis=1)

X=pd.concat([states,X],axis=1)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

from sklearn.metrics import r2\_score

score=r2\_score(y\_test,y\_pred)

print('Accuracy R2 Score',score)

import statsmodels.api as sm

X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)

X\_opt = X[:, [0, 1, 2, 3, 4, 5]]

regressor\_OLS = sm.OLS( y, X\_opt).fit()

print(regressor\_OLS.summary())

X\_opt = X[:, [0, 1, 3, 4, 5]]

regressor\_OLS = sm.OLS( y, X\_opt).fit()

print(regressor\_OLS.summary())

X\_opt = X[:, [0, 3, 4, 5]]

regressor\_OLS = sm.OLS( y, X\_opt).fit()

print(regressor\_OLS.summary())

X\_opt = X[:, [0, 3, 5]]

regressor\_OLS = sm.OLS(y, X\_opt).fit()

print(regressor\_OLS.summary())

X\_opt = X[:, [0, 3]]

regressor\_OLS = sm.OLS(endog = y, exog = X\_opt).fit()

print(regressor\_OLS.summary())

**Apriori Algorithm**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Market\_Basket\_Optimisation.csv', low\_memory=False, header=None)

!pip install apyori

list\_of\_transactions = []

for i in range(0, 7501):

list\_of\_transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

list\_of\_transactions[0]

from apyori import apriori

rules = apriori(list\_of\_transactions, min\_support = 0.004, min\_confidence = 0.2, min\_lift = 3, min\_length = 2)

results = list(rules)

def inspect(results):

lhs = [tuple(result [2] [0] [0]) [0] for result in results]

rhs = [tuple(result [2] [0] [1]) [0] for result in results]

supports = [result [1] for result in results]

confidences = [result [2] [0] [2] for result in results]

lifts = [result [2] [0] [3] for result in results]

return list(zip(lhs,rhs,supports,confidences, lifts))

resultsinDataFrame = pd.DataFrame(inspect(results),columns = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'] )

resultsinDataFrame.head(10)

**K-Means Clustering Algorithm**

from sklearn.datasets import load\_iris

from itertools import cycle

from sklearn.decomposition import PCA

from sklearn.cluster import KMeans

from numpy.random import RandomState

import pylab as pl

import matplotlib.pyplot as plt

class clustering:

def \_\_init\_\_(self):

self.plot(load\_iris().data)

def plot(self, X):

wcss=[]

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++',max\_iter=300,n\_init=10,random\_state=0)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.show()

pca = PCA(n\_components=2, whiten=True).fit(X)

X\_pca = pca.transform(X)

kmeans = KMeans(n\_clusters=3, random\_state=RandomState(42)).fit(X\_pca)

plot\_2D(X\_pca, kmeans.labels\_, ["c0", "c1", "c2"])

def plot\_2D(data, target, target\_names):

colors = cycle('rgbcmykw')

target\_ids = range(len(target\_names))

pl.figure()

for i, c, label in zip(target\_ids, colors, target\_names):

pl.scatter(data[target == i, 0], data[target == i, 1],

c=c, label=label)

pl.legend()

pl.show()

if \_\_name\_\_ == '\_\_main\_\_':

c = clustering()

**K Nearest Neighbour algorithm**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix of KNN \n",cm)

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('K-NN (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('K-NN (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

score = classifier.score(X\_test, y\_test)

print('Accuracy Score is: ',score)

**Hierarchical Clustering Algorithm**

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Euclidean distances')

plt.axhline(y=200, color='r', linestyle='--')

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y\_hc = hc.fit\_predict(X)

plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_hc == 1, 0], X[y\_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(X[y\_hc == 4, 0], X[y\_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**Random Forest Classification**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Random Forest Classification (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Random Forest Classification (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**CART algorithm**

from sklearn import tree

from sklearn.datasets import load\_iris

iris = load\_iris()

clf = tree.DecisionTreeClassifier()

clf = clf.fit(iris.data, iris.target)

import graphviz

dot\_data = tree.export\_graphviz(clf, out\_file=None)

graph = graphviz.Source(dot\_data)

graph.render("iriscart")

dot\_data = tree.export\_graphviz(clf, out\_file=None,

feature\_names=iris.feature\_names,

class\_names=iris.target\_names,

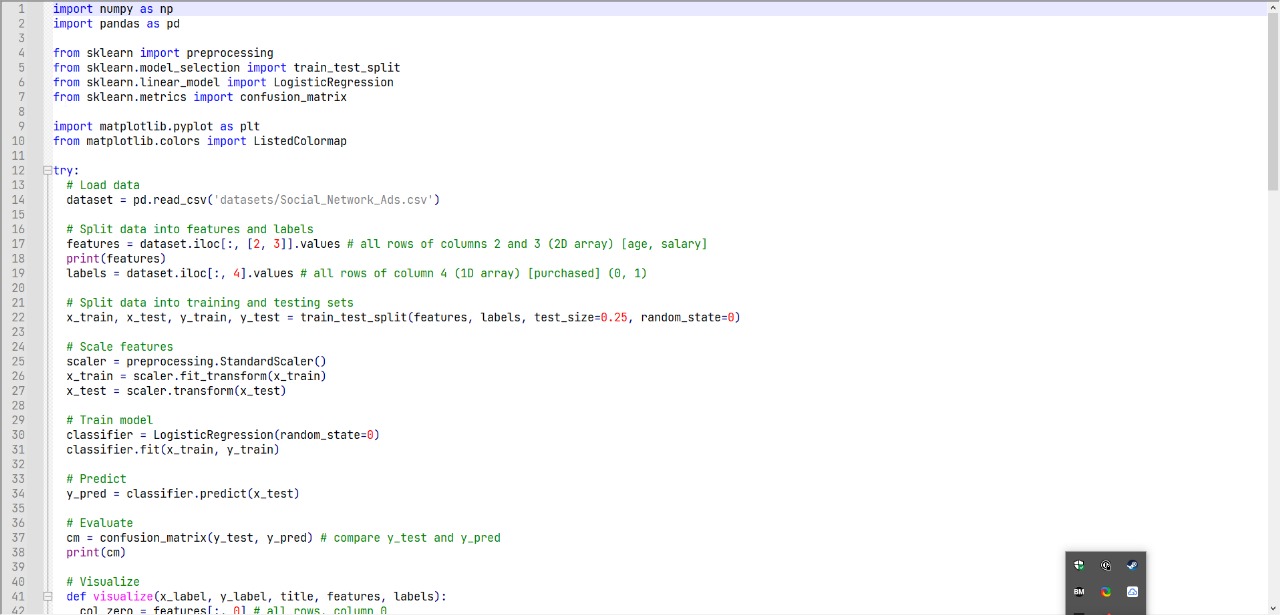
filled=True, rounded=True,

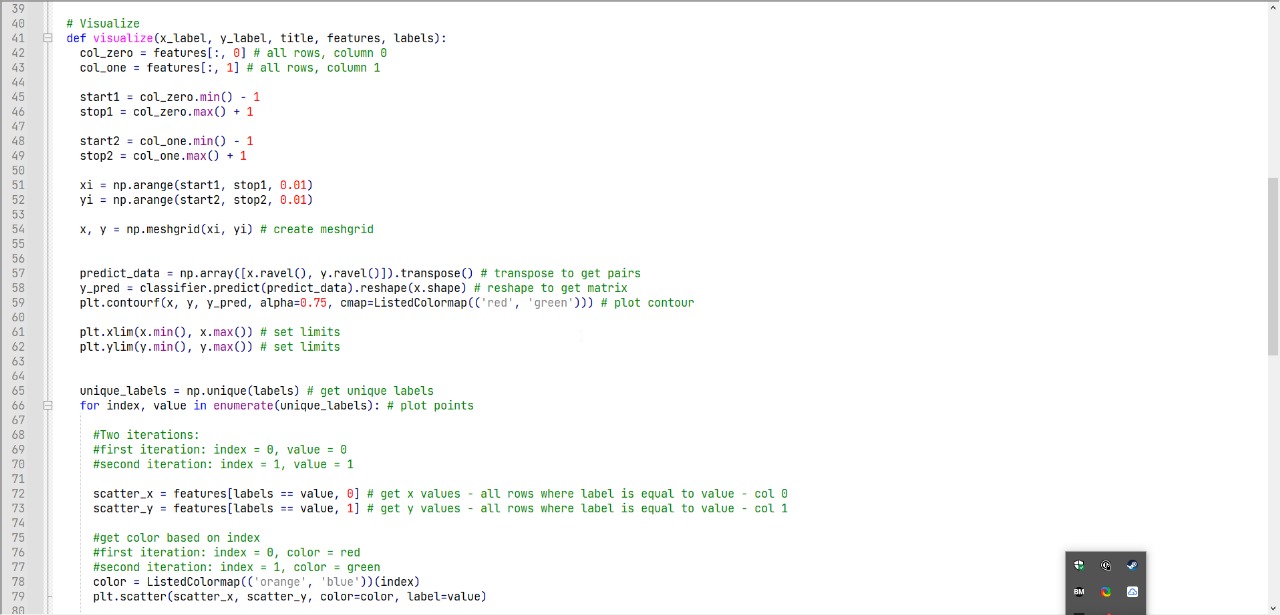
special\_characters=True)

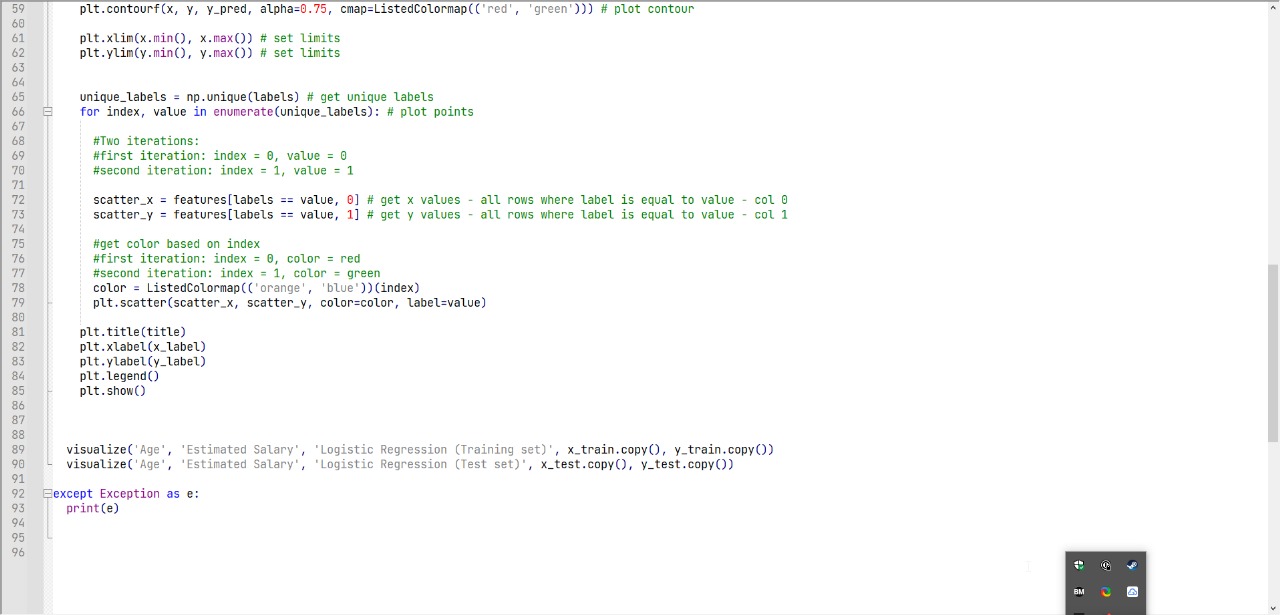
graph = graphviz.Source(dot\_data)

graph.view()

**Logistic regression using sigmoid function**

****

****

****

**Support vector machine algorithm using various kernel functions.**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Kernel SVM to the Training set

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

#classifier = SVC(kernel = 'linear', random\_state = 0)

#from sklearn.svm import LinearSVC

#classifier = LinearSVC( random\_state = 0)

#classifier = SVC(kernel = 'poly', random\_state = 0)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix of SVM \n",cm)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Kernel SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Kernel SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

score = classifier.score(X\_test, y\_test)

print('Accuracy Score is: ',score)