**Knearest-neighbours**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

x=ds.iloc[:,:-1].values

y=ds.iloc[:,-1].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)

print(classifier.predict(sc.transform([[30,87000]])))

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

print(np.concatenate(y\_pred.reshape(len(y\_pred),1),y\_test.reshape(len(y\_test),1)),1)

from sklearn.metrics import confusion\_matrix, accuracy\_score

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

accuracy\_score(y\_test,y\_pred)

from matplotlib.colors import ListedColormap

x\_set, y\_set= sc.inverser\_tranform(x\_train), y\_train

x1, x2 = np.meshgrid(np.arange(start=x\_set[:,0].min() -10, stop= x\_set[:,0].max()+10, step=1),

np.arange(start=x\_set[:,1].min()-1000, stop = x\_set[:, -1].max()+1000, step=1))

plt.contourf(x1, x2, classifier.predict(sc.transform(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('knn training set')

plt.xlabel('age')

plt.ylabel('estimated salary')

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

x\_set, y\_set= sc.inverser\_tranform(x\_test), y\_test

x1, x2 = np.meshgrid(np.arange(start=x\_set[:,0].min() -10, stop= x\_set[:,0].max()+10, step=1),

np.arange(start=x\_set[:,1].min()-1000, stop = x\_set[:, -1].max()+1000, step=1))

plt.contourf(x1, x2, classifier.predict(sc.transform(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('knn test set')

plt.xlabel('age')

plt.ylabel('estimated salary')

plt.legend()

plt.show()

**Linear Regression**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Salary\_Data.csv')

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

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

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

plt.scatter(X\_train, y\_train, color = 'red')

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

**Multiple Linear Regression**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

y = dataset.iloc[:, -1].values

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

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)

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

**KMeans**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

x=ds.iloc[:,[3,4]].values

from sklearn.cluster import KMeans

wcss=[]

for i in range(1,11):

kmeans=Kmeans(n\_clusters=i, init='k-means++', random\_state=42)

kmeans.fit(x)

wcss.append(kmeans.inertia\_)

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

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans=Kmeans(n\_clusters=5, init= 'k-means++', random\_state= 42)

y\_kmeans= kmeans.fit\_predict(x)

plt.scatter(x[y\_kmeans==0,0], x[y\_kmeans == 0,1], s=100, c='red' label='cluster 1')

plt.scatter(x[y\_kmeans==1,0], x[y\_kmeans== 1,1], s=100, c='blue', label='cluster 2')

plt.scatter(x[y\_kmeans==2,0], x[y\_kmeans == 2,1], s=100, c='green' label='cluster 3')

plt.scatter(x[y\_kmeans==3,0], x[y\_kmeans== 3,1], s=100, c='yellow', label='cluster 4')

plt.scatter(x[y\_kmeans==4,0], x[y\_kmeans== 4,1], s=100, c='cyan', label='cluster 4')

plt.scatter(kmeans.cluster\_centers\_[:,0], kmeans.cluster\_centers\_[:,1], s=300, c= 'yellow', label='Centroids')

plt.title('Clusters of customers')

plt.xlabel("Annual income k$")

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

plt.legend()

plt.show()

**Heirarchical Clustering**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

x=ds.iloc[:,[3,4]].values

import scipy.cluster.hierarchy as sch

dendogram = sch.dendogram(sch.linkage(x, method='ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('Eucidean distances')

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()

**ANN**

import numpy as np

import pandas as pd

import tensorflow as tf

ds= pd.read\_csv('Churn\_Modelling.csv')

x=ds.iloc[:,3:-1].values

y=ds.iloc[:,-1].values

from sklearn.preprocessing import LabelEncoder

le= LabelEncoder()

x[:,2]=le.fit\_transform(x[:,2])

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct= ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder="passthrough")

x=np.array(ct.fit\_transform(x))

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)

ann= tf.keras.models.Sequential()

ann.add(tf.keras.layers.Denser(units=6, activation='relu'))

ann.add(tf.keras.layers.Denser(units=6, activation = 'relu'))

ann.add(tf.keras.layers.Denser(units=1, activation='sigmoid'))

ann.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

ann.fit(x\_train, y\_train, batch\_size=32, epochs=100)

print(ann.predict(sc.transform([[1,0,0,0,600,1,40,3,60000,2,1,1,50000]]))> 0.5)

y\_pred=ann.predict(x\_test)

y\_pred=(y\_pred>0.5)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix, accuracy\_score

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

print(cm)

accuracy\_score(y\_test, y\_pred)

**SVM**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

x=ds.iloc[:,:-1].values

y=ds.iloc[:,-1].values

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size=0.5, 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.svm import svc

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

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

print(classifer.predict(sc.transform([[30,87000]])))

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

print(np.concatenate(y\_pred.reshape(len(y\_pred), 1), y\_test.reshape(len(y\_test),1)),1)

from sklearn.metrics import confusion\_matrix, accuracy\_score

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

accuracy\_score(y\_test, y\_pred)

from matplotlib.colors import ListedColormap

x\_set, y\_set = sc.inverse\_transform(x\_train), y\_train

x1, x2= np.meshgrid(np.arrange(start= x\_set[:,0].min() -10, stop= x\_set[:,0].max()+10,step=0.25), np.arange(start= x\_set[:,1].min() -1000, stop = x\_set[:,1].max()+1000, step=0.25))

plt.contourf(x1, x2, classifer.predict(sc.transform(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("SVM(Training Set)")

plt.xlabel("Age")

plt.ylabel("Estimated Salary")

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

x\_set, y\_set = sc.inverse\_transform(x\_test), y\_test

x1, x2= np.meshgrid(np.arrange(start= x\_set[:,0].min() -10, stop= x\_set[:,0].max()+10,step=0.25), np.arange(start= x\_set[:,1].min() -1000, stop = x\_set[:,1].max()+1000, step=0.25))

plt.contourf(x1, x2, classifer.predict(sc.transform(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("SVM(Test set)")

plt.xlabel("Age")

plt.ylabel("Estimated Salary")

plt.legend()

plt.show()