Practical 1 – Simple Linear Regression

* import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

dataset.head()

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

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

* print(X)
* print(y)
* 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)

* print(X\_train)
* print(X\_test)
* print(y\_train)
* print(y\_test)
* from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

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

y\_pred

* y\_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()

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

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

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

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

Practical 2 – Multiple Linear Regression

* import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('50\_Startups-2.csv')

dataset.head()

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

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

* print(X)
* print(y)
* 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))

* print(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))

Practical 3 – Logistic Regression

* import pandas as pd

iris\_data = pd.read\_csv('Iris.csv')

iris\_data.head()

* iris\_data.info()
* from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

iris\_data['Species'] = encoder.fit\_transform(iris\_data['Species'])

* iris\_data.head(150)
* import matplotlib.pyplot as plt

plt.pie(iris\_data['Species'].value\_counts(),labels=['Setosa','Versicolor','Virginica'],autopct='%0.2f')

plt.show()

* x = iris\_data.drop('Species',axis=1)

y=iris\_data['Species']

* print(x)
* print(y)
* 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=2)

* from sklearn.linear\_model import LogisticRegression

model = LogisticRegression(max\_iter=1000)

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

* pred\_train = model.predict(x\_train)
* from sklearn.metrics import confusion\_matrix,accuracy\_score

accuracy\_score(y\_train,pred\_train)

* pred\_test = model.predict(x\_test)
* accuracy\_score(y\_test,pred\_test)
* confusion\_matrix(y\_test,pred\_test)

Practical 4 – KNN

* import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

* dataset = pd.read\_csv('Social\_Network\_Ads.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=0.25,random\_state=0)

* print(X\_train)
* print(X\_test)
* print(y\_train)
* print(y\_test)
* from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

* print(X\_train)
* print(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([[40,200000]])))
* 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\_pred,y\_test)

print(cm)

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

Practical 5 – SVM

* import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

* dataset = pd.read\_csv('Social\_Network\_Ads.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=0.25,random\_state=0)

* print(X\_train)
* print(X\_test)
* print(y\_train)
* print(y\_test)
* from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

* print(X\_train)
* print(X\_test)
* from sklearn.svm import SVC

classifier = SVC(kernel=’linear’)

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

* print(classifier.predict(sc.transform([[40,200000]])))
* 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\_pred,y\_test)

print(cm)

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

Practical 6 – K-Means

* import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

print(X)

* from sklearn.cluster import KMeans

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

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

print(y\_kmeans)

* 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 ='cyan', label='Cluster 4')

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

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()

Practical 7 – Hierarchical Clustering

* import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

print(X)

* from sklearn.cluster import AgglomerativeClustering

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

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

print(y\_hc)

* plt.scatter(X[y\_hc == 0, 0], X[y\_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1') # plotting cluster 2

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

plt.scatter(X[y\_hc == 2, 0], X[y\_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3') # plotting cluster 4

plt.scatter(X[y\_hc == 3, 0], X[y\_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') # plotting cluster 5

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()

Practical 8 – ANN

* import numpy as np

import pandas as pd

import tensorflow as tf

* tf.\_\_version\_\_
* dataset = pd.read\_csv('Churn\_Modelling.csv')

X = dataset.iloc[:,3:-1].values

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

* print(X)
* print(y)
* from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

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

* print(X)
* 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))

* print(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.preprocessing import StandardScaler

sc= StandardScaler()

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

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

* ann = tf.keras.models.Sequential()
* ann.add(tf.keras.layers.Dense(units=6,activation='relu'))
* ann.add(tf.keras.layers.Dense(units=6,activation='relu'))
* ann.add(tf.keras.layers.Dense(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([(1,0,0,600,1,40,3,60000,2,1,1,50000)]))>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)

Practical 9 – CNN

* import tensorflow as tf

from keras.preprocessing.image import ImageDataGenerator

* tf.\_\_version\_\_
* train\_datagen = ImageDataGenerator(rescale=1./225,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip= True)

training\_set =train\_datagen.flow\_from\_directory('drive/MyDrive/small\_dataset/training\_set',

target\_size=(64,64),

batch\_size = 32,

class\_mode ='binary')

* test\_datagen = ImageDataGenerator(rescale = 1./255)

test\_set = test\_datagen.flow\_from\_directory('drive/MyDrive/small\_dataset/test\_set',

target\_size =(64,64),

batch\_size = 32,

class\_mode = 'binary')

* cnn = tf.keras.models.Sequential()
* cnn.add(tf.keras.layers.Convolution2D(filters=32,kernel\_size=3,activation='relu',input\_shape=[64,64,3]))
* cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2,strides=2))
* cnn.add(tf.keras.layers.Convolution2D(filters=32,kernel\_size=3,activation ='relu'))

cnn.add(tf.keras.layers.MaxPool2D(pool\_size=2,strides=2))

* cnn.add(tf.keras.layers.Flatten())
* cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))
* cnn.add(tf.keras.layers.Dense(units=1,activation='sigmoid'))
* cnn.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])
* cnn.fit(x=training\_set,validation\_data=test\_set,epochs=25)
* import numpy as np

from tensorflow.keras.preprocessing import image

test\_image = image.load\_img('drive/MyDrive/small\_dataset/single\_prediction/cat\_or\_dog\_1.jpg',target\_size=(64,64))

test\_image =image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image,axis=0)

result = cnn.predict(test\_image)

training\_set.class\_indices

if result[0][0]==1:

prediction ='dog'

else:

prediction='cat'

* print(prediction)