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.reshap
  e(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','Versicol
  or','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([(1,0,0,600,1,40,3,60000,2,1,1,500
  00)]))>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.reshap
  e(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/t
  raining_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/tes
  t_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)