**Supplementary Material for Session 05**

1. **Some important Python libraries and packages for Machine Learning**

In Python, there are many libraries and packages that make Machine Learning easier. Some of them are as below:

|  |  |
| --- | --- |
| *Libraries* | *pip command for installation* |
| [Numpy](https://pypi.org/project/numpy/) | pip install numpy |
| [Scipy](https://pypi.org/project/scipy/) | pip install scipy |
| [Scikit-learn](https://pypi.org/project/scikit-learn/) | pip install scikit-learn |
| [Pandas](https://pypi.org/project/pandas/) | pip install pandas |
| [Matplotlib](https://pypi.org/project/matplotlib/) | pip install matplotlib |
| [Seaborn](https://pypi.org/project/seaborn/) | pip install seaborn |
| [Plotly](https://pypi.org/project/plotly/) | pip install plotly |
| [NLTK](https://pypi.org/project/nltk/) | pip install nltk |
| [Scrapy](https://pypi.org/project/Scrapy3/) | pip install scrapy / pip install Scrapy3 |
| [TextBlob](https://pypi.org/project/textblob/) | pip install textblob |

The [Python Package Index (PyPI)](https://pypi.org/) is a repository of software for the Python programming language. pip is the package management system used to install and manage software packages written in Python.

1. **Reading and exploring a CSV file**

|  |
| --- |
| *# Import the necessary libraries*  import **matplotlib.pyplotasplot**  import **pandas**  *# Import the dataset*  dataset = pandas.read\_csv('salaryData.csv')  *#Explore the dataset*  print(dataset.shape) *# number of rows and columns*  print(dataset.head(5)) *# display first five rows of the dataset*  *# iterating the columns*  **for** col **in** dataset.columns:  print(col)  *# Differentiate attribute and target columns*  x = dataset['YearsExperience'].values  y = dataset['Salary'].values  *# print(x.shape) # shape of x*  *# print(y.shape) # shape of y*  X = x.reshape(len(x),1)  Y = y.reshape(len(y),1)  *# print(X.shape) # shape of X*  *# print(Y.shape) # shape of Y*  *# Plot the data*  plot.scatter(X, Y, color='RED')  plot.show() |

1. **Example of Linear Regression using Python is as below.**

|  |
| --- |
| *# Import the necessary libraries*  import**numpy**  import**matplotlib.pyplotasplot**  import**pandas**  from**sklearnimport**metrics  from**sklearn.model\_selectionimport**train\_test\_split  from**sklearn.linear\_modelimport**LinearRegression  *# Import the dataset*  dataset=pandas.read\_csv('salaryData.csv')  *#Explore the dataset*  print(dataset.shape)*# number of rows and columns*  print(dataset.head(5))*# display first five rows of the dataset*  *# Differentiate attribute and target columns*  x=dataset['YearsExperience'].values  y=dataset['Salary'].values  *# print(x.shape) # shape of x*  *# print(y.shape) # shape of y*  X = x.reshape(len(x),1)  Y = y.reshape(len(y),1)  *# print(X.shape) # shape of X*  *# print(Y.shape) # shape of Y*  *# Split the dataset into the training set and test set*  *# We're splitting the data in 1/3, so out of 30 rows, 20 rows will go into the training set, and 10 rows will go into the testing set.*  xTrain,xTest,yTrain,yTest=train\_test\_split(X,Y,test\_size=1/3)  *# Creating a LinearRegression object and fitting it on our training set.*  linearRegressor=LinearRegression()  linearRegressor.fit(xTrain,yTrain)  *# Predicting the test set results*  yPrediction=linearRegressor.predict(xTest)  *# Visualization*  df=pandas.DataFrame({'Actual':yTest.flatten(),'Predicted':yPrediction.flatten()})  print(df)  df1=df  df1.plot(kind='bar')  plot.grid(which='major',color='green')  plot.grid(which='minor',color='black')  plot.show()  plot.scatter(xTest,yTest,color='gray')  plot.plot(xTest,yPrediction,color='red',linewidth=2)  plot.show()  *# Displaying errors*  print('Mean Absolute Error:',metrics.mean\_absolute\_error(yTest,yPrediction))  print('Mean Squared Error:',metrics.mean\_squared\_error(yTest,yPrediction))  print('Root Mean Squared Error:',numpy.sqrt(metrics.mean\_squared\_error(yTest,yPrediction))) |

1. **Example of k-Nearest Neighbor classifier using Scikit Learn in Python is as below**

|  |
| --- |
| *# importing necessary libraries*  fromsklearnimportdatasets  fromsklearn.metricsimportconfusion\_matrix  fromsklearn.model\_selectionimporttrain\_test\_split  *# loading the iris dataset*  iris=datasets.load\_iris()  *# X -> features, y -> label*  X=iris.data  y=iris.target  *# dividing X, y into train and test data*  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.67)  *# training a KNN classifier*  fromsklearn.neighborsimportKNeighborsClassifier  knn=KNeighborsClassifier(n\_neighbors=7).fit(X\_train,y\_train)  *# accuracy on X\_test*  accuracy=knn.score(X\_test,y\_test)  print(accuracy)  *# creating a confusion matrix*  knn\_predictions=knn.predict(X\_test)  print(knn\_predictions)  cm=confusion\_matrix(y\_test,knn\_predictions)  print(cm) |

1. **Decision Tree**

|  |
| --- |
| Example of Decision Tree Classifier |
| *# importing necessary libraries*  from **sklearnimport** datasets  from **sklearn.metricsimport** confusion\_matrix  from **sklearn.model\_selectionimport** train\_test\_split  from **pandasimport** DataFrame  from **sklearnimport** tree  from **sklearn.metricsimport** accuracy\_score  *# loading the iris dataset*  iris = datasets.load\_iris()  *# X -> features, Y -> label*  X = iris.data  Y = iris.target  *# print(X)*  *# print(Y)*  *# dividing X, Y into train and test data*  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=1/3)  *# training a decision tree classifier*  model = tree.DecisionTreeClassifier()  model.fit(X\_train, Y\_train)  *# testing*  model\_predictions = model.predict(X\_test)  *# accuracy of prediction*  accuracyScore = accuracy\_score(Y\_test, model\_predictions)  print(accuracyScore) |
|  |
| Regression with Decision Tree |
| *# Import the necessary libraries*  import**numpy**  import**matplotlib.pyplotasplot**  import**pandas**  from**sklearnimport**metrics  from**sklearn.model\_selectionimport**train\_test\_split  from**sklearn.treeimport**DecisionTreeRegressor  *# Import the dataset*  dataset=pandas.read\_csv('salaryData.csv')  *# Differentiate attribute and target columns*  x=dataset['YearsExperience'].values  y=dataset['Salary'].values  X=x.reshape(len(x),1)  Y=y.reshape(len(y),1)  *# Split the dataset into the training set and test set*  xTrain,xTest,yTrain,yTest=train\_test\_split(X,Y,test\_size=1/3)  *# Creating a DecisionTreeRegressor on our trainging set.*  regressor=DecisionTreeRegressor()  regressor.fit(xTrain,yTrain)  *# Predicting the test set results*  yPrediction=regressor.predict(xTest)  *# Displaying errors*  print('Mean Absolute Error:',metrics.mean\_absolute\_error(yTest,yPrediction)) |

1. **Naïve Bayes Classifier**

|  |
| --- |
| *# importing necessary libraries*  from**sklearnimport**datasets  from**sklearn.metricsimport**confusion\_matrix  from**sklearn.model\_selectionimport**train\_test\_split  from**pandasimport**DataFrame  from**sklearn.naive\_bayesimport**GaussianNB  from**sklearn.metricsimport**accuracy\_score  *# loading the wine dataset*  wine=datasets.load\_wine()  *# X -> features, Y -> label*  X=wine.data  Y=wine.target  *# print(X)*  *# print(Y)*  *# dividing X, Y into train and test data*  X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=1/3)  *# training a Gaussian Naive Bayes classifier*  model=GaussianNB()  model.fit(X\_train,Y\_train)  *# testing*  model\_predictions=model.predict(X\_test)  *# accuracy of prediction*  accuracyScore=accuracy\_score(Y\_test,model\_predictions)  print(accuracyScore) |

1. **Neural Network Models**

|  |
| --- |
| Classification with Neural Network |
| import **warnings**  warnings.simplefilter("ignore")  import **pandasaspd**  wine = pd.read\_csv('wine\_data.csv', names = ["Cultivator", "Alchol", "Malic\_Acid", "Ash", "Alcalinity\_of\_Ash", "Magnesium", "Total\_phenols", "Falvanoids", "Nonflavanoid\_phenols", "Proanthocyanins", "Color\_intensity", "Hue", "OD280", "Proline"])  *# original dataset link: https://archive.ics.uci.edu/ml/datasets/Wine*  print(wine.head())  print(wine.shape)  X = wine.drop('Cultivator',axis=1)  Y = wine['Cultivator']  from **sklearn.model\_selectionimport** train\_test\_split  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=1/3)  from **sklearn.preprocessingimport** StandardScaler  scaler = StandardScaler()  scaler.fit(X\_train)  X\_train = scaler.transform(X\_train)  X\_test = scaler.transform(X\_test)  from **sklearn.neural\_networkimport** MLPClassifier  mlp = MLPClassifier()  mlp.fit(X\_train,Y\_train)  predictions = mlp.predict(X\_test)  from **sklearn.metricsimport** accuracy\_score  accuracyScore = accuracy\_score(Y\_test, predictions)  print(accuracyScore) |

1. **An example of k-means Clustering using Scikit-Learn is as below.**

|  |
| --- |
| from pandasimport DataFrame  import matplotlib.pyplotasplt  from sklearn.clusterimport KMeans  Data = {'x': [25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],  'y': [79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]  }  df = DataFrame(Data,columns=['x','y'])  plt.title("Data before clustering")  plt.scatter(df['x'], df['y'])  plt.show()  cn =int(input("How many clusters? :"))  kmeans = KMeans(n\_clusters=cn).fit(df)  centroids = kmeans.cluster\_centers\_  print(centroids)  plt.title("Data after clustering")  plt.scatter(df['x'], df['y'], c= kmeans.labels\_)  plt.scatter(centroids[:, 0], centroids[:, 1], c='red')  plt.show() |

1. **Cross Validation**

|  |
| --- |
| from **sklearn.datasetsimport** load\_iris  from **sklearn.neighborsimport** KNeighborsClassifier  from **sklearn.model\_selectionimport** KFold  from **sklearn.metricsimport** accuracy\_score  import **warnings**  warnings.simplefilter("ignore")  iris\_data = load\_iris()  x = iris\_data.data  y = iris\_data.target  kf = KFold(n\_splits=5)  print("**\n**K nearest neighbor:")  **for** train\_index, test\_index **in** kf.split(x):  x\_train, x\_test = x[train\_index], x[test\_index]  y\_train, y\_test = y[train\_index], y[test\_index]  knn = KNeighborsClassifier()  knn.fit(x\_train,y\_train)  prediction = knn.predict(x\_test)  print(accuracy\_score(y\_test, prediction)) |