**Input-Output in data:**

* age - age in years
* sex - (1 = male; 0 = female)
* cp - chest pain type
* trestbps - resting blood pressure (in mm Hg on admission to the hospital)
* chol - serum cholestoral in mg/dl
* fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
* restecg - resting electrocardiographic results
* thalach - maximum heart rate achieved
* exang - exercise induced angina (1 = yes; 0 = no)
* oldpeak - ST depression induced by exercise relative to rest
* slope - the slope of the peak exercise ST segment
* ca - number of major vessels (0-3) colored by flourosopy
* thal - 3 = normal; 6 = fixed defect; 7 = reversable defect
* target - have disease or not (1=yes, 0=no)

**Inputs:**

* age - age in years
* sex - (1 = male; 0 = female)
* cp - chest pain type
* trestbps - resting blood pressure (in mm Hg on admission to the hospital)
* chol - serum cholestoral in mg/dl
* fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
* restecg - resting electrocardiographic results
* thalach - maximum heart rate achieved
* exang - exercise induced angina (1 = yes; 0 = no)
* oldpeak - ST depression induced by exercise relative to rest
* slope - the slope of the peak exercise ST segment
* ca - number of major vessels (0-3) colored by flourosopy
* thal - 3 = normal; 6 = fixed defect; 7 = reversable defect

**Output:**

* target - have disease or not (1=yes, 0=no)

**SLIDE:**

**Heart-Disease Prediction System**

* **Real-World Problem:**
  + **Heart-Disease Prediction**
* **Treated as:**
  + **Supervised Machine Learning Problem**
* **Note:**
  + **Heart-Disease Prediction Problem is treated as a**
    - **Binary Classification Problem because**
      * **The Main AIM is to Distinguish between Two Classes**
        + **Class 01 = Have Disease (1)**
        + **Class 02 = Not Have Disease (0)**
* **Goal:**
  + **Learn an Input-Output Function**
    - **Learn from Input to Predict Output**

**SLIDE:**

**Heart-Disease Prediction System – TASK:**

* **Given:**
  + **A Patient (Represented as Set of Attributes)**
* **Task:**
  + **Automatically Predict whether the Patient have Heart Disease or Not.**

**SLIDE:**

**Heart-Disease Prediction System – TASK:**

* **Input:**
  + **A Patient**
* **Output:**
  + **Have Heart-Disease/Not Have Heart-Disease.**

**SLIDE:**

* **In Kaggle Heart-Disease Dataset, A Patient is represented with many attributes**
* **Kaggle Heart-Disease Dataset:**
* [**https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset?resource=download**](https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset?resource=download)
* **For Simplicity and to explain things more clearly** 
  + **In this, Lecture, we have represented a Patient with Five Attributes.**

**SLIDE:**

**Heart-Disease Predication System – Input Attributes:**

* **In this lecture, a Patient is represented with the following Five Attributes**
* **Attributes**
* Age
* Sex
* Cp
* chol

**SLIDE:**

**Heart-Disease Prediction System – Output Attributes:**

* **In Heart-Disease Dataset, there is One Output Attribute**
  + **Attribute 05 – Target:**
    - Possible Value 01 = Yes (1)
    - Possible Value 02 = No (0)

**SLIDE:**

**Heart-Disease Prediction System – Summary (Input and Output)**

* **The Following table summarizes the Input and Output Attributes for Heart-Disease Dataset**

**SLIDE:**

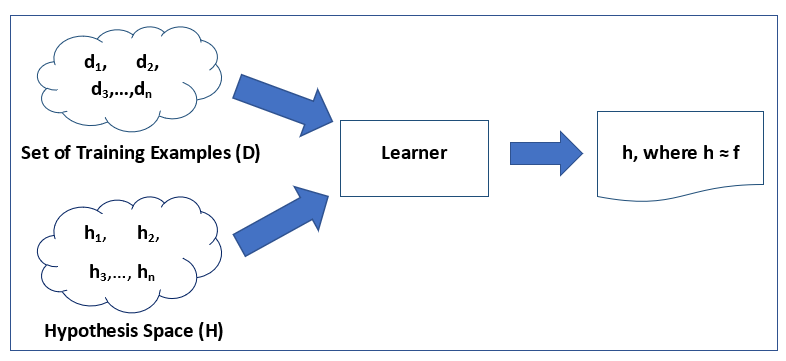
**Heart-Disease Prediction System:**

* **Task**
  + **Develop a Heart-Disease Prediction System to Predict the Disease of a Patient.**
* **Input**
  + **4 attributes as mentioned above**

* **Output**
  + **One Attribute**

|  |
| --- |
| 1. **Yes** |

* **Treated as a**
  + **Supervised Machine Learning Problem**
* **Goal**
  + **Learn an Input-Output Function**
    - **Learn form Input to predict the Output**



**SLIDE**

**Learning Input-Output Function – General Settings Cont…**

* **Input to Learner**
  + **Set of Training Examples (D)**
  + **Set of Hypothesis (a.k.a. Hypothesis Space (H))**
* **Job of Learner**
  + **The main job of a Learner is to search the Hypothesis Space (H) using the Set of Training Examples (D) to find out a Hypothesis (h) from Hypothesis Space (H), which best fits the Set of Training Examples (D)**
* **Output of Learner**
  + **A Learner outputs a Hypothesis (h) from Hypothesis Space (H), which best fits the Set of Training Examples (D)**

**SLIDE**

**Learning Input-Output Function – General Settings Conti…**

* **To summarize**
  + **Learning is a Searching Problem**

|  |
| --- |
| **Dataset** |

**SLIDE**

**Dataset**

* **TheDataset (or Sample Data), used for thisLecturecomprisesof** 
  + **100 Instances**
    - **See heart-disease-sample-data.csv File in theData and Code**
* **Sample Data Characteristics** 
  + **Total Instances in Sample Data = 100**
    - **Have Disease = 50**
    - **Not Have Disease = 50**
* **Note**
  + **Forsimplicityandexplain things more clearly, we have used a** 
    - **Small Dataset**
* **Remember** 
  + **TocompletelyandcorrectlyunderstandanyReal-world Task**
    - **Step 1:First executeit ata small leveli.e.Prototype Level**
    - **Step 2:Executethe Real-world Task atFull Scale**
  + **If youcannot executeandunderstanda Real-world Task atPrototype LevelThen**
    - **Youcannot executeandunderstandit atFull Scale😊**

|  |
| --- |
| **Technique** |

**SLIDE**

**Machine Learning Algorithm – Support Vector Classifier (SVC)**

* **Forany Machine Learning Problem, you need to know the following main things**
  1. **Representation of Training Examples**
  2. **Representation of Hypothesis**
  3. **Searching Strategy**
  4. **Training Regime**
  5. **Main Parameters**
  6. **Implementation**

**SLIDE**

**Representation of Training Examples**

* **For the Support Vector Classifier (SVC)Machine Learning Algorithm,Training Exampleisrepresentedas**
  + **Attribute-Value Pair**
* **RepresentationofInput**
  + **Numeric**
* **RepresentationofOutput**
  + **Numeric**

**SLIDE**

**Representation of Hypothesis (h)**

* **InMachine Learning,Representation of Hypothesis (h) may vary from Machine Learning Algorithm to Machine Learning Algorithm**
  + **In this Lecture, Machine Learning AlgorithmSupport Vector Classifier (SVC)is used**
* **Representation of Hypothesis (h)** 
  + **I am not clear about the Representation of Hypothesis in SVM. Please drop me an email if you know 😊Jazak Allah Khair**
* **Hypothesis Space (H)**
  + **Set of Hypothesis (h)**

**SLIDE**

**Searching Strategy**

* **InSupport Vector Classifier (SVC),Searching Strategyis**
  + **Ranking Strategy**
    - **One-Versus-All Strategy**

**SLIDE**

**Training Regime**

* **Inthe Support Vector Classifier (SVC),Training Regimeis**
  + **Incremental Method**

**SLIDE**

**Implementation**

* **In this Lecture, weimplementedtheSupport Vector Classifier (SVC)using**
  + **Python** 
    - **Version 3.7.4**
  + **Jupyter Notebook** 
    - **Version 6.0.1**
  + **Scikit-Learn Machine Learning Toolkit** 
    - **Version0.21.2**

|  |
| --- |
| **Evaluation Methodology** |

**SLIDE**

**Evaluation Methodology**

* **The problem ofHeart Disease Prediction istreatedas a** 
  + **Supervised Machine LearningTask**
* **Supervised Machine Learningis treated as a Binary Classification Task**
  + **The aim is to distinguish between Two Classes**

|  |
| --- |
| * **Class / Category /Label 01** * **0 (No)** * **Class / Category /Label 02**   + **1 (Yes)** |

**SLIDE**

**Evaluation Methodology Cont….**

* **We will use a Train-Test Split Ratio of 80-20 to estimate the performance of the Support Vector Classifier (SVC)**
* **We Split the Sample Data using**
  + **Train-Test Split Ratio of**
    - **80% - 20%**
* **Sample Data** 
  + **Total Instances = 100**
    - **Survived = 50**
    - **Not Survived = 50**
* **Training Data** 
  + **Total Instances = 80**
    - **Survived = 40**
    - **Not Survived = 40**
* **Testing Data** 
  + **Total Instances = 20**
    - **Survived = 10**
    - **Not Survived = 10**

|  |
| --- |
| **Evaluation Measure** |

**SLIDE**

**Evaluation Measure**

* **In this Lecture, Evaluationis carried out using** 
  + **Accuracy**

**SLIDE**

**Accuracy**

* **Definition**
  + **Accuracy is defined as the proportion of correctly classified Test Instances**
* **Formula**

|  |
| --- |
|  |

* **Note**
  + **Error = 1 –Accuracy**

|  |
| --- |
| **Coding Setup** |

**SLIDE**

**Coding Setup**

* **In this Section, we will present**
  + **System Settings**
  + **Libraries**
  + **Built-in Functions**
  + **User-Defined Functions**
  + **Basic Terms**
  + **Variable Names**

|  |
| --- |
| **System Settings** |

**SLIDE**

**Libraries**

* **In this Lecture, I used the following Libraries to Write Code for**
  + **Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

|  |  |
| --- | --- |
| **Pandas** | |
| **Definition** | **Pandas is a software library written for the Python Programming Language for Data Manipulation and Analysis that runs on top of Numpy** |
| **Purpose** | **Used for Data Science and Data Analytics** |
| **Documentation Link** | [**https://pandas.pydata.org/docs/**](https://pandas.pydata.org/docs/) |

|  |  |
| --- | --- |
| **NumPy** | |
| **Definition** | **NumPy is a general-purpose array-processing package** |
| **Purpose** | **Numpy provides**   * **High-performance multidimensional array** * **Tools to compute with and manipulate these arrays** |
| **Documentation Link** | **https://numpy.org/doc/** |

|  |  |
| --- | --- |
| **Pickle** | |
| **Definition** | **The pickle module implements binary protocols for serializing and de-serializing a Python object structure** |
| **Purpose** | **Pickling is the process whereby a Python object hierarchy is converted into a byte stream** |
| **Documentation Link** | [**https://docs.python.org/3/library/pickle.html**](https://docs.python.org/3/library/pickle.html) |

|  |  |
| --- | --- |
| **LabelEncoder** | |
| **Definition** | **LabelEncoder is a utility class to help normalize labels such that they contain only values between 0 and n\_classes-1** |
| **Purpose** | **Encode categorical features as a one-hot numeric array** |
| **Documentation Link** | [**https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html**](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html) |

|  |  |
| --- | --- |
| **SVM** | |
| **Definition** | **Support vector machines (SVMs) are a set of supervised learningmethodsusedfor**[**classification**](https://scikit-learn.org/stable/modules/svm.html#svm-classification)**,**[**regression**](https://scikit-learn.org/stable/modules/svm.html#svm-regression)**, and**[**outlier detection**](https://scikit-learn.org/stable/modules/svm.html#svm-outlier-detection)**.** |
| **Purpose** | **The main objective is to segregate the given dataset in the best possible way. The distance between thenearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset.** |
| **Documentation Link** | [**https://scikit-learn.org/stable/modules/svm.html**](https://scikit-learn.org/stable/modules/svm.html) |

|  |  |
| --- | --- |
| **PrettyTable** | |
| **Definition** | **PrettyTable is a simple Python library designed to make it quick and easy to represent tabular data in visually appealing ASCII tables** |
| **Purpose** | **A simple Python library for easily displaying tabular data in a visually appealing ASCII table format** |
| **Documentation Link** | [**https://pypi.org/project/PrettyTable/**](https://pypi.org/project/PrettyTable/) |

|  |  |
| --- | --- |
| **Accuracy\_Score** | |
| **Definition** | **Accuracy is defined as the proportion of correctly classified Test Instances.** |
| **Purpose** | **CalculateAccuracyScore.** |
| **Documentation Link** | [**https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\_score.html**](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html) |

|  |  |
| --- | --- |
| **train\_test\_split** | |
| **Definition** | **train\_test\_splitis a function in Sklearn model selection for splitting data arrays into two subsets: for training data and testing data.** |
| **Purpose** | **Split arrays or matrices into the random train and test subsets.** |
| **Documentation Link** | **https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html** |

**SLIDE**

**Built-in Functions**

* **In this Lecture, I used the following Built-in Functions to Write Code for**
  + **Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

|  |  |
| --- | --- |
| **Function 01** | |
| **Function Name** | **read\_csv()** |
| **Purpose** | **To Read a CSV File in Pandas DataFrame** |

|  |  |
| --- | --- |
| **Function 02** | |
| **Function Name** | **to\_csv()** |
| **Purpose** | **Exports the DataFrame to CSV Format** |

|  |  |
| --- | --- |
| **Function 03** | |
| **Function Name** | **fit()** |
| **Purpose** | **Used to Train the Data** |

|  |  |
| --- | --- |
| **Function 04** | |
| **Function Name** | **transform()** |
| **Purpose** | **To Transforms the Data** |

|  |  |
| --- | --- |
| **Function 05** | |
| **Function Name** | **iloc()** |
| **Purpose** | **To Select the Specific Columns and Rows from Dataframe** |

|  |  |
| --- | --- |
| **Function 06** | |
| **Function Name** | **pandas.set\_option()** |
| **Purpose** | **Sets the value of the specified option** |

|  |  |
| --- | --- |
| **Function 07** | |
| **Function Name** | **accuracy\_score()** |
| **Purpose** | **Compute Accuracy Score** |

|  |  |
| --- | --- |
| **Function 08** | |
| **Function Name** | **Predict()** |
| **Purpose** | **Given a trained model, Predict the label of a new set of Data** |

|  |  |
| --- | --- |
| **Function 09** | |
| **Function Name** | **score()** |
| **Purpose** | **Returns the Accuracy Score of the Trained Model** |
|  | |
| **Function 10** | |
| **Function Name** | **dump()** |
| **Purpose** | **Used to store objects in a file** |

|  |  |
| --- | --- |
| **Function 11** | |
| **Function Name** | **load()** |
| **Purpose** | **To retrieve Pickled Data** |

|  |  |
| --- | --- |
| **Function 12** | |
| **Function Name** | **add\_row()** |
| **Purpose** | **Used to add Rows in a Pretty Table** |

|  |  |
| --- | --- |
| **Function 13** | |
| **Function Name** | **PrettyTable()** |
| **Purpose** | **Represent tabular data in visually Appealing Tables** |

|  |  |
| --- | --- |
| **Function 14** | |
| **Function Name** | **np.ravel()** |
| **Purpose** | **Used to create a contiguous Flattened Array** |

|  |  |
| --- | --- |
| **Function 15** | |
| **Function Name** | **train\_test\_split()** |
| **Purpose** | **Split sample data into training data and testing data** |

**SLIDE**

**Machine Learning Cycle**

* **Four phases of a Machine Learning Cycle are**
  + **Training Phase**
    - **Build the Model using Training Data**
  + **Testing Phase**
    - **Evaluate the performance of Model using Testing Data**
  + **Application Phase**
    - **Deploy the Model in the Real-world, to predict Real-time unseen Data**
  + **Feedback Phase**
    - **Take Feedback from the Users and Domain Experts to improve the Model**

|  |
| --- |
| **Steps – Executing Machine Learning Cycle Using a Single File** |

**SLIDE**

**Steps – Executing Machine Learning Cycle Using a Single File**

* **In Sha Allah, we will follow the following steps to execute the Machine Learning Cycle Using a Single File**
  + **Step 1: Import Libraries**
  + **Step 2: LoadSample Data**
  + **Step 3: Understand and Pre-processSample Data**
    - **Step 3.1: UnderstandSample Data**
    - **Step 3.2: Pre-process Sample Data**
  + **Step 4:Feature Extraction**
  + **Step 5: Label Encoding (Input and Output is converted in Numeric Representation)**
    - **Step 5.1: Train the Label Encoder**
    - **Step 5.2: Label Encode the Output**
    - **Step 5.3: Label Encode the Input**
  + **Step 6: Execute the Training Phase**
    - **Step 6.1: Splitting Sample Data into Training Data and Testing Data**
    - **Step 6.2: Splitting Input Vectorsand Outputs/Labels of Training Data**
    - **Step 6.3: Train the Support Vector Classifier**
    - **Step 6.4: Save the Trained Model**
  + **Step 7: Execute the Testing Phase**
    - **Step 7.1: Splitting Input Vectors and Output/Labels of Testing Data**
    - **Step 7.2: Load the Saved Model**
    - **Step 7.3: Evaluate the Performance of TrainedModel**
      * **Step 7.3.1: Make Predictions from the Model on Testing Data**
    - **Step 7.4: Calculate the Accuracy Score**
  + **Step 8: Execute the Application Phase**
    - **Step 8.1: Take Input from User**
    - **Step 8.2: Convert User Input into Feature Vector (Exactly Same as Feature Vectors of Sample Data)**
    - **Step 8.3: Label Encoding of Feature Vector (Exactly Same as Label Encoded Feature Vectors of Sample Data)**
    - **Step 8.4: Load the Saved Model**
    - **Step 8.5: Model Prediction**
      * **Apply Model on the Label Encoded Feature Vector of unseen instance and return Prediction to the User**
  + **Step 9: Execute the Feedback Phase**
  + **Step 10: Improve the Model based on Feedback**

***CODING SECTION***

**Importing Libraries:**

import numpy as np

import pandas as pd

import pickle

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

from sklearn.preprocessing import LabelEncoder

from sklearn import svm

from sklearn.metrics import accuracy\_score

from astropy.table import Table, Column

**Load Sample Data**

sample\_data = pd.read\_csv("")

print("\n\nSample Data:")

print("============\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data)

**Understand Sample Data**

print("\n\nAttributes in Sample Data:")

print("==========================\n")

print(sample\_data.columns)

print("\n\nNumber of Instances in Sample Data:",sample\_data["age"].count())

print("========================================\n")

sample\_data\_encoded\_output = sample\_data.copy()

original\_sample\_data = sample\_data.copy()

**Labels**

age = pd.DataFrame({"age":[0,1,2]})

sex = pd.DataFrame({"sex":[0,1]})

cp = pd.DataFrame({"cp":[0,1,2,3]})

chol = pd.DataFrame({"chol":[0,1]})

target = pd.DataFrame({"target":[1,0]})

**Initialize the Label Encoders**

age\_label\_encoder = LabelEncoder()

sex\_label\_encoder = LabelEncoder()

cp\_label\_encoder = LabelEncoder()

chol\_label\_encoder = LabelEncoder()

target\_label\_encoder = LabelEncoder()

**Train the Label Encoders**

age\_label\_encoder.fit(np.ravel(age))

sex\_label\_encoder.fit(np.ravel(sex))

cp\_label\_encoder.fit(np.ravel(cp))

chol\_label\_encoder.fit(np.ravel(chol))

target\_label\_encoder.fit(np.ravel(target))

**Transform Output of into Numerical Representation**

print("\n\nSurvived Attribute After Label Encoding:")

print("========================================\n")

sample\_data["encoded\_survived"] = target\_label\_encoder.transform(sample\_data['target'])

print(sample\_data[["target", "encoded\_survived"]])

**Print Original and Encoded Ouput Sample Data**

sample\_data\_encoded\_output[['age', 'sex', 'cp', 'chol', 'target']] = sample\_data[['age', 'sex', 'cp', 'chol', 'target']]

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print("\n\nOriginal Sample Data:")

print("=====================\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(original\_sample\_data)

print("\n\nSample Data after Label Encoding of Output:")

print("===========================================\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded\_output)

**Save the Transformed Features into CSV File**

sample\_data\_encoded\_output.to\_csv(r'sample-data-encoded-output.csv', index = False, header = True)

print("\n\age Attribute After Label Encoding:")

print("======================================\n")

sample\_data\_encoded\_output["encoded\_age"] = age\_label\_encoder.transform(sample\_data\_encoded\_output['age'])

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded\_output[["age", "encoded\_age"]])

print("\n\nsex Attribute After Label Encoding:")

print("======================================\n")

sample\_data\_encoded\_output["encoded\_sex"] = sex\_label\_encoder.transform(sample\_data\_encoded\_output['sex'])

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded\_output[["sex", "encoded\_sex"]])

print("\n\ncp Attribute After Label Encoding:")

print("=======================================\n")

sample\_data\_encoded\_output["encoded\_cp"] = cp\_label\_encoder.transform(sample\_data\_encoded\_output['cp'])

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded\_output[["cp", "encoded\_cp"]])

print("\n\nchol Attribute After Label Encoding:")

print("========================================\n")

sample\_data\_encoded\_output["encoded\_chol"] = target\_label\_encoder.transform(sample\_data\_encoded\_output['chol'])

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded\_output[["chol", "encoded\_chol"]])

**Print Original and Encoded Sample Data**

sample\_data\_encoded[['age', 'sex', 'cp', 'chol', 'target']] = sample\_data\_encoded\_output[['age', 'sex', 'cp', 'chol', 'target']]

print("\n\nOriginal Sample Data:")

print("=====================\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(original\_sample\_data)

print("\n\nSample Data after Label Encoding:")

print("=================================\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(sample\_data\_encoded)

**Save the Transformed Features into CSV File**

sample\_data\_encoded.to\_csv(r'sample-data-encoded.csv', index = False, header = True)

training\_data\_encoded, testing\_data\_encoded = train\_test\_split( sample\_data\_encoded , test\_size=0.2 , random\_state=0 , shuffle = False)

**Save the Training and Testing Data into CSV File**

training\_data\_encoded.to\_csv(r'training-data-encoded.csv', index = False, header = True)

testing\_data\_encoded.to\_csv(r'testing-data-encoded.csv', index = False, header = True)

**Print Training and Testing Data**

print("\n\nTraining Data:")

print("==============\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(training\_data\_encoded)

print("\n\nTesting Data:")

print("==============\n")

pd.set\_option("display.max\_rows", None, "display.max\_columns", None)

print(testing\_data\_encoded)

print("\n\nInputs Vectors (Feature Vectors) of Training Data:")

print("==================================================\n")

input\_vector\_train = training\_data\_encoded.iloc[: , :-1]

print(input\_vector\_train)

print("\n\nOutputs/Labels of Training Data:")

print("================================\n")

print(" Survived")

output\_label\_train = training\_data\_encoded.iloc[: ,-1]

print(output\_label\_train)

**Train the Support Vector Classifier**

print("\n\nTraining the Support Vector Classifier on Training Data")

print("========================================================\n")

print("\nParameters and their values:")

print("============================\n")

svc\_model = svm.SVC(gamma='auto',random\_state=0)

svc\_model.fit(input\_vector\_train,np.ravel(output\_label\_train))

print(svc\_model)

**Save the Model in a Pkl File**

pickle.dump(svc\_model, open('svc\_trained\_model.pkl', 'wb'))

print("\n\nInputs Vectors (Feature Vectors) of Testing Data:")

print("=================================================\n")

input\_vector\_test = testing\_data\_encoded.iloc[: , :-1]

print(input\_vector\_test)

print("\n\nOutputs/Labels of Testing Data:")

print("==============================\n")

print(" Survived")

output\_label\_test = testing\_data\_encoded.iloc[: ,-1]

print(output\_label\_test)

**Load the Saved Model**

model = pickle.load(open('svc\_trained\_model.pkl', 'rb'))

**Provide Test data to the Trained Model**

model\_predictions = model.predict(input\_vector\_test)

testing\_data\_encoded.copy(deep=True)

pd.options.mode.chained\_assignment = None

testing\_data\_encoded["Predictions"] = model\_predictions

**Save the Predictions into CSV File**

testing\_data\_encoded.to\_csv(r'model-predictions.csv', index = False, header = True)

model\_predictions = testing\_data\_encoded

print("\n\nPredictions Returned by svc\_trained\_model:")

print("==========================================\n")

print(model\_predictions)

**Calculate the Accuracy**

model\_accuracy\_score = accuracy\_score(model\_predictions["target"],model\_predictions["Predictions"])

print("\n\nAccuracy Score:")

print("===============\n")

print(round(model\_accuracy\_score,2))