AI 2024 Online Summer Internship

Titanic Project

Lecture 2

Developing a System for Titanic Passenger Survival Prediction using Train-Test Split Approach

Authors:

Mr. Hamza Safdar

Mr. Waqar Ashiq

Dr. Rao Muhammad Adeel

Lecture Outline

- **Introduction**
- **Experimental Setup**
- **Coding Setup**
- **Titanic Passenger Survival Prediction System using Train-Test Split Approach – Machine Learning Cycle**

Introduction

SLIDE

Royal Mail Ship (RMS) Titanic – Brief Overview

- **RMS Titanic was a British passenger liner operated by the White Star Line that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City**
- **Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking one of modern history's deadliest peacetime commercial marine disasters**
- **RMS Titanic was the largest ship afloat at the time she entered service[1]**

SLIDE

RMS Titanic – Main Features

- **Name of Ship**
  - **Royal Mail Ship (RMS) Titanic**
- **Manufactured**
  - **May 31, 1911**
- **Size**
  - **Length = 882 feet 9 inches (269.06 m)**
  - **Width = 92.5 feet (28.2 m)**
  - **Height = 104 feet (32 m)**
- **First Journey**
  - **April 10, 1912**
- **Source and Destination**
  - **Southampton, England to New York, USA**
- **Total Passengers on Board**
  - **2,208**
- **Passengers Survived**
  - **705**
- **Passengers Died**
  - **1,503**

[1] Last visited: 07-09-2020

SLIDE
RMS Titanic

Figure 01: Royal Mail Ship (RMS) Titanic [Source]

SLIDE
Lecture Focus

- **The main focus of this Lecture is developing a**
  - **Predictive System which can automatically predict whether a Passenger in RMS Titanic Ship Survived or Not**

SLIDE
Titanic Passenger Survival Prediction System

- **Real-world World**
  - **Titanic Passenger Survival**
- **Treated as**
  - **Supervised Machine Learning Problem**
- **Note**
  - **Titanic Passenger Survival Prediction Problem is treated as a**
    - **Binary Classification Problem because the**
      - **The main aim is to distinguish between Two Classes**

- Class 01 = Survived
- Class 02 = Not Survived

- Goal
  - Learn an Input-Output Function
    - i.e. Learn from Input to predict the Output

SLIDE
Titanic Passenger Survival Prediction System – Task

- Given
  - A Passenger (Represented as Set of Attributes)
- Task
  - Automatically predict whether the Passenger Survived or Not

SLIDE
Titanic Passenger Survival Prediction System – Input and Output

- Input
  - A Passenger
- Output
  - Survived / Not Survived

SLIDE
Note

- In Kaggle Titanic Dataset, a Passenger is represented with many Attributes
- Kaggle Titanic Dataset
  - URL:
- For simplicity and to explain things more clearly
  - In this Lecture, we have represented a Passenger with Four Attributes

SLIDE
Titanic Passenger Survival Prediction System – Input Attributes

- In this Lecture, a Passenger is represented with the following Four Attributes
- Attribute 01 – PClass
  - Possible Value 01 = First
  - Possible Value 02 = Second

- Possible Value 03 = Third
- Attribute 02 – Gender
  - Possible Value 01 = Female
  - Possible Value 02 = Male
- Attribute 03 – Sibling
  - Possible Value 01 = Zero
  - Possible Value 02 = One
  - Possible Value 03 = Two
  - Possible Value 04 = Three
- Attribute 04 – Embarked
  - Possible Value 01 = Cherbourg
  - Possible Value 02 = Southampton
  - Possible Value 03 = Queenstown

SLIDE

Titanic Passenger Survival Prediction System – Output Attributes

- In Titanic Dataset, there is One Output Attribute
  - Attribute 01 – Survived
    - Possible Value 01 = Yes
    - Possible Value 02 = No

SLIDE

Titanic Passenger Survival Prediction System – Summary (Input and Output)

- The following Table Summarizes the Input and Output Attributes for Titanic Dataset

Attribute No.	Attribute Names	Possible Values	Data Types
1	PClass	First, Second, Third	Categorical
2	Gender	Male, Female	Categorical
3	Sibling	Zero, One, Two, Three	Categorical

4	Embarked	Southampton, Queenstown, Cherbourg	Categorical
5	Survived	Yes, No	Categorical

Table 01: Attributes of Dataset

SLIDE

Titanic Passenger Survival Prediction Problem – Treated as

- The problem of Titanic Passenger Survival Prediction using Titanic Dataset is treated as a
  - Binary Classification Problem
- Reason
  - There are Two possible Output Values for each instance

SLIDE

Learning Input-Output Function – General Settings

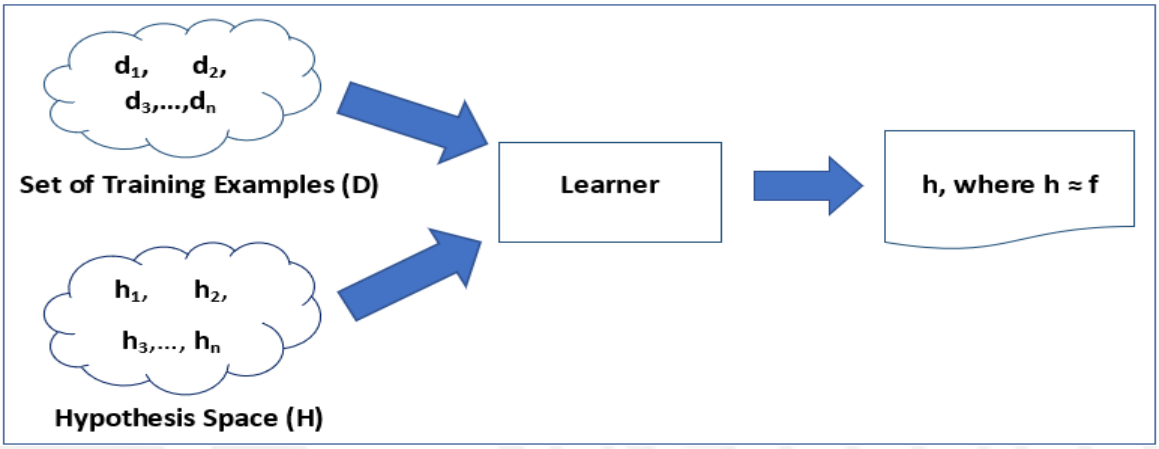

Figure 02: Learning Input-Output Function – General Settings

- Note
  - In the above Figure
    - Learner refers to a Machine Learning Algorithm

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

Experimental Setup

SLIDE

Experimental Setup

- **The four main components of an Experimental Setup are**
  1. **Dataset**
  2. **Technique**
  3. **Evaluation Methodology**
  4. **Evaluation Measures**

Dataset

SLIDE

Dataset

- **The Dataset (or Sample Data), used for this Lecture comprises of**
  - **100 Instances**
    - **See sample-data.csv File in the Data and Code**
- **Sample Data Characteristics**
  - **Total Instances in Sample Data = 100**
    - **Survived = 50**
    - **Not Survived = 50**
- **Note**
  - **For simplicity and explain things more clearly, we have used a**
    - **Small Dataset**
- **Remember**
  - **To completely and correctly understand any Real-world Task**
    - **Step 1: First execute it at a small level i.e. Prototype Level**
    - **Step 2: Execute the Real-world Task at Full Scale**
  - **If you cannot execute and understand a Real-world Task at Prototype Level Then**
    - **You cannot execute and understand it at Full Scale** 😊

Techniqu e

SLIDE

Machine Learning Algorithm – Support Vector Classifier (SVC)

- **For any 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 Example is represented as**
  - **Attribute-Value Pair**
- **Representation of Input**
  - **Numeric**
- **Representation of Output**
  - **Numeric**

SLIDE

Representation of Hypothesis (h)

- **In Machine Learning, Representation of Hypothesis (h) may vary from Machine Learning Algorithm to Machine Learning Algorithm**
  - **In this Lecture, Machine Learning Algorithm Support 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

- In Support Vector Classifier (SVC), Searching Strategy is
  - Ranking Strategy
    - One-Versus-All Strategy

SLIDE

Training Regime

- In the Support Vector Classifier (SVC), Training Regime is
  - Incremental Method
- For more details on Training Regimes
  - See Lecture – Basics of Machine Learning
  - URL:

SLIDE

Main Parameters – Support Vector Classifier (SVC)

- Important Parameters to consider in designing Support Vector Classifier (SVC) are as follows
- C:
  - Regularization Parameter
  - Value will be
    - float
  - Default Value for C
    - 1.0
- Kernel:
  - Specifies the kernel type to be used in the algorithm
  - kernel Possible Values
    - linear
    - poly
    - rbf
    - sigmod
    - precomputed
  - Default Value for kernel
    - rbf
- degree:
  - Degree of the polynomial kernel function (poly)

- Value will be
  - Int
- Default Value for degree
  - 3

- Gamma:
  - Kernel coefficient for rbf, poly, and sigmoid.
  - Value will be
    - scale, auto or float
  - Default value for Gamma is
    - scale
- cache_size:
  - Specify the size of the kernel cache (in MB).
  - Value will be
    - float
  - Default value for cache_size is
    - 200
- class_weight:
  - Set the parameter C of class i to class_weight[i]*C for SVC.
  - Value will be
    - dict or balanced
  - Default value for class_weight is
    - None
- verbose:
  - Enable verbose output
  - Value will be
    - bool
  - Default value for verbose is
    - False
- max_iter:
  - Hard limit on iterations within solver, or -1 for no limit.
  - Value will be
    - int
  - Default Value for max_iter
    - -1
- decision_function_shape:

- Value will be
  - ovo
  - ovr
- Default Value for decision_function_shape is
  - ovr

- break_ties:
  - predict will break ties according to the confidence values
  - Value will be
    - bool
  - Default value for break_ties is
    - False
- random_state:
  - Controls the pseudo-random number generation for shuffling the data for probability estimates.
  - Value will be
    - int or RandomState instance
  - Default value for random_state is
    - None

SLIDE
Note

- In this Lecture, we have used the Default Values for various Parameters of Support Vector Classifier (SVC) Algorithm
  - C = 1.0
  - cache_size = 200
  - class_weight = None
  - decision_functiom_shape = ovr
  - degree = 3
  - gamma = auto
  - kernel = rbf
  - max_iter = -1
  - random_state = 0
  - verbose = False

SLIDE
Implementation

- In this Lecture, we implemented the Support Vector Classifier (SVC) using
  - Python
    - Version 3.7.4
  - Jupyter Notebook
    - Version 6.0.1
  - Scikit-Learn Machine Learning Toolkit
    - Version 0.21.2

Evaluation Methodology

SLIDE

Evaluation Methodology

- The problem of Titanic Passenger Survival Prediction is treated as a
  - Supervised Machine Learning Task
- Supervised Machine Learning is 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, Evaluation is carried out using**
  - **Accuracy**

SLIDE

Accuracy

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

$$Accuracy = \frac{Correctly\ Classified\ Test\ Instances}{Total\ Number\ of\ Test\ Instances}$$

- **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

System Settings	
Developer Name	**Mr. Hamza Safdar**
Programming Language	**Python 3.7.4**
IDE	**Jupyter Notebook 6.0.1**
Machine Learning Toolkit	**Scikit Learn 0.21.2**
Code Version	**1.0**
Date	**28 – August – 2020**

Libraries

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	

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	

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	

SVM	
Definition	Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression, and outlier detection.
Purpose	The main objective is to segregate the given dataset in the best possible way. The distance between the nearest 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	

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	

Accuracy_Score	
Definition	**Accuracy is defined as the proportion of correctly classified Test Instances.**
Purpose	**Calculate Accuracy Score.**
Documentation Link	

train_test_split	
Definition	**train_test_split is 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	

SLIDE

Note

- **In Sha Allah, in the next Slides, I will try to explain the**
  - **Purpose of various**
    - **Built-in Functions used in the Project Titled: Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

Built-in Functions

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

Note

- **In Sha Allah, in the next Slides, I will try to explain the**
  - **Purpose, Arguments, and Return Type of various**
    - **User-Defined Functions used in the Project Titled: Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

User-Defined Functions

SLIDE

User-Defined Functions

- **In this Lecture, I have not used any User Defined Functions to Write Code for**
  - **Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

SLIDE

Note

- **In Sha Allah, in the next Slides, I will try to explain the**
  - **Name and Style of**

- **Basic Terms used in the Project Titled: Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

Basic Terms

SLIDE

Basic Terms

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

Basic Terms	
Sample Data	Numerical Input Values
Training Data	Numerical Output Values
Testing Data	Output Labels
Survived	Machine Learning Algorithms
Not Survived	Survival
Training Data Encoded	Predicted Survival
Testing Data Encoded	Label Encoding

SLIDE

Note

- **In Sha Allah, in the next Slides, I will try to explain the**
  - **Name and Style of**
  - **Variables used in the Project Titled: Developing a Titanic Passenger Survival Prediction System using Train-Test Split Approach**

Variable Names

SLIDE

Variable Names

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

Variable Names	
sample_data	user_input
sample_data_encoded_output	training_data_encoded
sample_data_encoded	testing_data_encoded
pclass	input_vector_train
gender	output_label_train
sibling	svc_model
embarked	input_vector_test
survived	output_label_test
pclass_label_encoder	model
gender_label_encoder	model_predictions
sibling_label_encoder	model_accuracy_score
embarked_label_encoder	pclass_input
survived_label_encoder	gender_input
unseen_data_features	sibling_input
predicted_survival	embarked_input

Titanic Passenger Survival Prediction System using Train-Test Split Approach – Machine Learning Cycle

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

SLIDE

Executing Machine Learning Cycle

- In Sha Allah, in this Section, we will execute the Machine Learning Cycle
  - Using a Single File
- Code
  - See Titanic Passenger Survival Prediction System using Train-Test Split Approach.ipynb File in Data and Code
- Note
  - Below Code does not contain Output
  - In Titanic Passenger Survival Prediction System using Train-Test Split Approach.ipynb File I have also shown Output of Code

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: Load Sample Data**
  - **Step 3: Understand and Pre-process Sample Data**
    - **Step 3.1: Understand Sample 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 Vectors and 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 Trained Model**
      - **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

SLIDE
Coding Section

Author Details

```
'''
*----------------------------- AUTHOR_DETAILS ------------------------------*
|                                                                           |
|        Project Title  =  Titanic Passenger Survival Prediction System     |
|                                                                           |
|        Author         =  Mr. Hamza Safdar                                 |
|                                                                           |
|        Copyright      =  Copyright (C) 2020 Mr. Hamza Safdar              |
|                                                                           |
|        License        =  Public Domain                                    |
|                                                                           |
|        Version        =  1.0                                              |
|                                                                           |
*---------------------------------------------------------------------------*
'''
```

SLIDE
Project Purpose

Project Purpose

---------- PROJECT PURPOSE ----------

The main purpose of this Project is to demonstrate how the Titanic Passenger Survival Problem can be treated as a Supervised Machine Learning Problem using Python and Scikit-learn Machine Learning Toolkit

For this Purpose, In Sha Allah, we will execute the Machine Learning Cycle

---

SLIDE

Step 1: Import Libraries

Import Libraries

```
# Import 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 prettytable import PrettyTable
from astropy.table import Table, Column
```

SLIDE

Step 2: Load Sample Data

Load Sample Data

```
# Load Sample Data
```

```
'''
*---------------------- LOAD_SAMPLE_DATA ------------------------*
|     Function: read_csv()                                       |
|             Purpose: Read a dataset in CSV file format         |
|     Arguments:                                                 |
|             path: Path to dataset file                         |
|             dataset: Dataset file name                         |
|     Return:                                                    |
|             dataset: Dataset in DataFrame format               |
*----------------------------------------------------------------*
'''

sample_data = pd.read_csv("sample-data.csv")

print("\n\nSample Data:")
print("============\n")
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(sample_data)
```

SLIDE

Step 3: Understand and Pre-process Sample Data

- **Step 3.1: Understand Sample Data**

Understand 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["PClass"].count())
print("=========================================\n")
```

SLIDE

Step 3: Understand and Pre-process Sample Data

- **Step 3.2: Pre-process Sample Data**
  - **Sample Data is already Pre-processed**

SLIDE

Step 4: Feature Extraction

- **The sample is already Featured Extracted**
  - **No Feature Extraction needs to be Performed**

SLIDE

Step 5: Label Encoding (Input and Output is converted in Numeric Representation)

- **Step 5.1: Train the Label Encoder**

Train the Label Encoder

```
# Train the Label Encoder

'''
*------------------ TRAIN_LABEL_ENCODER --------------------*
|        Function: Fit()                                    |
|              Purpose: Fit or Train the Label Encoder      |
|        Arguments:                                         |
|               Labels: Target Values                       |
|        Return:                                            |
|               Instance: Returns an instance of self       |
*-----------------------------------------------------------*
'''

# Labels

pclass = pd.DataFrame({"Pclass":["First","Second","Third"]})
gender = pd.DataFrame({"Gender":["Male","Female"]})
sibling = pd.DataFrame({"Sibling":["Zero","One","Two","Three"]})
embarked = pd.DataFrame({"Embarked":["Southampton","Cherbourg","Queenstown"]})
survived = pd.DataFrame({"Survived":["Yes","No"]})

# Initialize the Label Encoders

pclass_label_encoder = LabelEncoder()
gender_label_encoder = LabelEncoder()
sibling_label_encoder = LabelEncoder()
embarked_label_encoder = LabelEncoder()
survived_label_encoder = LabelEncoder()

# Train the Label Encoders

pclass_label_encoder.fit(np.ravel(pclass))
```

```
gender_label_encoder.fit(np.ravel(gender))
sibling_label_encoder.fit(np.ravel(sibling))
embarked_label_encoder.fit(np.ravel(embarked))
survived_label_encoder.fit(np.ravel(survived))
```

SLIDE

Step 5: Label Encoding (Input and Output is converted in Numeric Representation)

- **Step 5.2: Label Encode the Output**

Label Encode the Output

```
# Label Encoding of the Output

'''
*------------------ LABEL_ENCODE_OUTPUT --------------------*
|        Function: Transform()                              |
|              Purpose: Transform Input (Categorical)       |
|                       into Numerical Representation       |
|        Arguments:                                         |
|              Attribute: Target values                     |
|        Return:                                            |
|              Attribute: Numerical Representation          |
*-----------------------------------------------------------*
'''

sample_data_encoded_output = sample_data.copy()
original_sample_data = sample_data.copy()

# Transform Output of into Numerical Representation

print("\n\nSurvived Attribute After Label Encoding:")
print("=========================================\n")
sample_data["encoded_survived"] = survived_label_encoder.transform(sample_data
['Survived'])
print(sample_data[["Survived", "encoded_survived"]])

# Print Original and Encoded Ouput Sample Data

sample_data_encoded_output[['PClass', 'Gender', 'Sibling', 'Embarked', 'Surviv
ed']] = sample_data[['PClass', 'Gender', 'Sibling', 'Embarked', 'encoded_survi
ved']]
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 = F
alse, header = True)
```

SLIDE

Step 5: Label Encoding (Input and Output is converted in Numeric Representation)

- **Step 5.3: Label Encode the Input**

Label Encode the Input

```
# Label Encoding of the Input

'''
*------------------- LABEL_ENCODE_INPUT --------------------*
|        Function: Transform()                               |
|              Purpose: Transform Input (Categorical)        |
|                          into Numerical Representation     |
|        Arguments:                                          |
|              Attribute: Target values                      |
|        Return:                                             |
|              Attribute: Numerical Representation           |
*------------------------------------------------------------*
'''

sample_data_encoded = sample_data_encoded_output.copy()
sample_data_encoded_output_orignal = sample_data_encoded_output.copy()

# Transform Input Attributes into Numerical Representation

print("\n\nPClass Attribute After Label Encoding:")
print("======================================\n")
sample_data_encoded_output["encoded_pclass"] = pclass_label_encoder.transform(
sample_data_encoded_output['PClass'])
pd.set_option("display.max_rows", None, "display.max_columns", None)
```

```
print(sample_data_encoded_output[["PClass", "encoded_pclass"]])

print("\n\nGender Attribute After Label Encoding:")
print("=====================================\n")
sample_data_encoded_output["encoded_gender"] = gender_label_encoder.transform(
sample_data_encoded_output['Gender'])
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(sample_data_encoded_output[["Gender", "encoded_gender"]])

print("\n\nSibling Attribute After Label Encoding:")
print("======================================\n")
sample_data_encoded_output["encoded_sibling"] = sibling_label_encoder.transfor
m(sample_data_encoded_output['Sibling'])
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(sample_data_encoded_output[["Sibling", "encoded_sibling"]])

print("\n\nEmbarked Attribute After Label Encoding:")
print("=======================================\n")
sample_data_encoded_output["encoded_embarked"] = embarked_label_encoder.transf
orm(sample_data_encoded_output['Embarked'])
pd.set_option("display.max_rows", None, "display.max_columns", None)
print(sample_data_encoded_output[["Embarked", "encoded_embarked"]])

# Print Original and Encoded Sample Data

sample_data_encoded[['PClass', 'Gender', 'Sibling', 'Embarked', 'Survived']] =
sample_data_encoded_output[['encoded_pclass', 'encoded_gender', 'encoded_sibli
ng', 'encoded_embarked', 'Survived']]
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)
```

SLIDE

Step 6: Execute the Training Phase

- **Step 6.1: Splitting Sample Data into Training Data and Testing Data**

Splitting Sample Data into Training Data and Testing Data

```
# Splitting Sample Data into Training Data and Testing Data

'''
*------------------- SPLIT_SAMPLE_DATA ---------------------*
|        Function: train_test_split()                       |
|              Purpose: Split arrays or matrices into       |
|                       random train and test subsets       |
|        Arguments:                                         |
|              arrays: sequence of indexables               |
|              test_size: float or int                      |
|        Return:                                            |
|              splitting: list                              |
*-----------------------------------------------------------*
'''

training_data_encoded, testing_data_encoded = train_test_split( sample_data_en
coded , 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, head
er = 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)
```

SLIDE

Step 6: Execute the Training Phase

- **Step 6.2: Splitting Input Vectors and Outputs/Labels of Training Data**

Splitting Input Vectors, Outputs/Labels of Training Data

```
# Splitting Input Vectors and Outputs / Labels of Training Data

'''
*---------------- SPLIT_INPUT_VECTORS_AND_LABELS --------------*
|        Function: iloc()                                      |
|            Purpose: Splitting Input Vector and Labels        |
|        Arguments:                                            |
|            Attribute: Name or Location Attribute to Split    |
|        Return:                                               |
|            Attribute: Split Attributes                       |
*--------------------------------------------------------------*
'''

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)
```

SLIDE

Step 6: Execute the Training Phase

- **Step 6.3: Train the Support Vector Classifier**

Train the Support Vector Classifier

```
# Train the Support Vector Classifier

'''
*--------------- TRAIN_SUPPORT_VECTOR_CLASSIFIER ------------------*
|       Function: svm.SVC()                                       |
|           Purpose: Train the Algorithm on Training Data         |
|       Arguments:                                                |
|           Training Data: Provide Training Data to the Model     |
|       Return:                                                   |
|           Parameter: Model return the Training Parameters       |
*-----------------------------------------------------------------*
'''

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)
```

SLIDE

Step 6: Execute the Training Phase

- **Step 6.4: Save the Trained Model**

Save the Trained Model

```
# Save the Trained Model

'''
*--------------------- SAVE_THE_TRAINED_MODEL ----------------------*
|        Function: dump()                                           |
|             Purpose: Save the Trained Model on your Hard Disk     |
|        Arguments:                                                 |
|             Model: Model Objects                                  |
|        Return:                                                    |
|             File: Trained Model will be Saved on Hard Disk        |
*-------------------------------------------------------------------*
'''

# Save the Model in a Pkl File

pickle.dump(svc_model, open('svc_trained_model.pkl', 'wb'))
```

SLIDE

Step 7: Execute the Testing Phase

- **Step 7.1: Splitting Input Vectors and Outputs/Labels of Testing Data**

Splitting Input Vectors and Outputs/Labels of Testing Data

```
# Splitting Input Vectors and Outputs/Labels of Testing Data

'''
*---------------- SPLIT_INPUT_VECTORS_AND_LABELS --------------*
|        Function: iloc()                                      |
|            Purpose: Splitting Input Vector and Labels        |
```

```
|        Arguments:                                                  |
|            Attribute: Name or Location Attribute to Split          |
|        Return:                                                     |
|            Attribute: Split Attributes                             |
*--------------------------------------------------------------------*
'''

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)
```

SLIDE

Step 7: Execute the Testing Phase

- **Step 7.2: Load the Saved Model**

Load the Saved Model

```
# Load the Saved Model

'''
*------------------- LOAD_SAVED_MODEL --------------------------*
|         Function: load()                                      |
|               Purpose: Method to Load Previously Saved Model  |
|         Arguments:                                            |
|               Model: Trained Model                            |
|         Return:                                               |
|               File: Saved Model will be Loaded in Memory      |
*---------------------------------------------------------------*
'''

# Load the Saved Model

model = pickle.load(open('svc_trained_model.pkl', 'rb'))
```

SLIDE

Step 7: Execute the Testing Phase

- **Step 7.3: Evaluate the Machine Learning Model**
  - **Step 7.3.1: Make Predictions with the Trained Models on Testing Data**

Evaluate the Machine Learning Model

```
# Evaluate the Machine Learning Model

'''
*--------------------- EVALUATE_MACHINE_LEARNING_MODEL ----------------------*
|       Function: Predict()                                                  |
|            Purpose: Make a Prediction using Algorithm on Test Data         |
|       Arguments:                                                           |
|            Testing Data: Provide Test data to the Trained Model            |
|       Return:                                                              |
|            Predictions: Model return Predictions                           |
*----------------------------------------------------------------------------*
'''

# 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)
```

SLIDE

Step 7: Execute the Testing Phase

- **Step 7.4: Calculate the Accuracy Score**

Calculate the Accuracy Score

```
# Calculate the Accuracy Score

'''
/*----------------------- CALCULATE_ACCURACY_SCORE -------------------*
|          Function: accuracy_score()                                 |
|                Purpose: Evaluate the algorithm on Testing data      |
|          Arguments:                                                 |
|                Prediction: Predicted values                         |
|                Label: Actual values                                 |
|          Return:                                                    |
|                Accuracy: Accuracy Score                             |
*---------------------------------------------------------------------*
'''

# Calculate the Accuracy

model_accuracy_score = accuracy_score(model_predictions["Survived"],model_pred
ictions["Predictions"])

print("\n\nAccuracy Score:")
print("===============\n")
print(round(model_accuracy_score,2))
```

SLIDE

Step 8: Execute the Application Phase

- **Step 8.1: Take Input from User**

Take Input from User

```
# Take Input from User

'''
*---------------- TAKE_USER_INPUT ----------------*
'''

pclass_input = input("\nPlease enter PClass here (First,Second,Third) : ").str
ip()
gender_input = input("\nPlease enter your Gender here (Male, Female) : ").stri
p()
sibling_input = input("\nPlease enter your Sibling here (Zero,One,Two,Three) :
").strip()
```

```
embarked_input = input("\nPlease enter Embarked here (Cherbourg,Southampton,Queenstown) : ").strip()
```

SLIDE

Step 8: Execute the Application Phase

- **Step 8.2: Convert User Input into Feature Vector (Exactly Same as Feature Vectors of Sample Data)**

Convert User Input into Feature Vector

```
# Convert User Input into Feature Vector

user_input = pd.DataFrame({ 'PClass': [pclass_input],'Gender': [gender_input], 'Sibling': [sibling_input],'Embarked': [embarked_input]})

print("\n\nUser Input Feature Vector:")
print("==========================\n")
print(user_input)
```

SLIDE

Step 8: Execute the Application Phase

- **Step 8.3: Label Encoding of Feature Vector (Exactly Same as Label Encoded Feature Vectors of Sample Data)**

Label Encoding of Feature Vector

```
# Label Encoding

'''
*----------------- TRANSFORM_UNSEEN_INTPUT_FEATURES ---------------*
|           Function: Transform()                                  |
|                 Purpose: Transform Input (Categorical) into      |
|                          Numerical Representation                |
|           Arguments:                                             |
|                 Attribute: Target values                         |
|           Return:                                                |
|                 Attribute: Numerical Representation              |
*------------------------------------------------------------------*
'''
```

```
# Transform Input (Categorical) Attributes of Unseen Data into Numerical Repre
sentation

unseen_data_features = user_input.copy()
unseen_data_features["PClass"] = pclass_label_encoder.transform(user_input['PC
lass'])
unseen_data_features["Gender"] = gender_label_encoder.transform(user_input['Ge
nder'])
unseen_data_features["Sibling"] = sibling_label_encoder.transform(user_input['
Sibling'])
unseen_data_features["Embarked"] = embarked_label_encoder.transform(user_input
['Embarked'])

print("\n\nUser Input Feature Vector:")
print("==========================\n")
print(user_input)

print("\n\nUser Input Encoded Feature Vector:")
print("==================================\n")
print(unseen_data_features)
```

SLIDE

Step 8: Execute the Application Phase

- **Step 8.4: Load the Saved Model**

Load the Saved Model

```
# Load the Saved Model

'''
*---------------------- LOAD_SAVED_MODEL -------------------------*
|         Function: load()                                       |
|             Purpose: Method to Load Previously Saved Model     |
|         Arguments:                                             |
|               Model: Trained Model                             |
|         Return:                                                |
|               File: Saved Model will be Loaded in Memory       |
*----------------------------------------------------------------*
'''

# Load the Saved Model
```

```
model = pickle.load(open('svc_trained_model.pkl', 'rb'))
```

SLIDE

Step 8: Execute the Application Phase

- **Step 8.5: Model Prediction**
  - **Step 8.5.1: Apply Model on the Label Encoded Feature Vector of unseen instance and return Prediction to the User**

Model Prediction

```
# Prediction of Unseen Instance

'''
*---------------------------  MODEL_PREDICTION --------------------------*
|           Function: predict()                                          |
|                 Purpose: Use Trained Model to Predict the Output       |
|                          of Unseen Instances                           |
|           Arguments:                                                   |
|                 User Data: Label Encoded Feature Vector of             |
|                            Unseen Instances                            |
|           Return:                                                      |
|                 Survival: Survived or Not Survived                     |
*------------------------------------------------------------------------*
'''

# Make a Prediction on Unseen Data

predicted_survival = model.predict(unseen_data_features)

if(predicted_survival == 1):
    prediction = "SURVIVED"
if(predicted_survival == 0):
    prediction = "NOT SURVIVED"

# Add the Prediction in a Pretty Table

pretty_table = PrettyTable()
pretty_table.add_column("       ** Prediction **       ",[prediction])
print(pretty_table)
```

SLIDE

Step 9: Execute the Feedback Phase

- A Two-Step Process
  - Step 01: After some time, take Feedback from
    - Domain Experts and Users on deployed Titanic Passenger Survival Prediction System
  - Step 02: Make a List of Possible Improvements based on Feedback received

SLIDE

Step 10: Improve Model based on Feedback

- There is Always Room for Improvement
- Based on Feedback from Domain Experts and Users
  - Improve your Model

TODO and Your Turn

SLIDE
TODO

- **Task**
  - **Consider the Heart Disease Classification Problem. The main aim is to predict whether a patient has Heart Disease or Not (i.e. Binary Classification Problem)?**
  - **Heart Disease Dataset Link**
    - **URL:**
  - **For simplicity, I have taken a sample of 100 instances from the Original Heart Disease Dataset**
    - **See heart-disease-sample-data.csv File in Supporting Material**
- **Note**
  - **Your answer should be**
    - **Well Justified**
- **Question**
  - **Write down the Input and Output of the Heart Disease Classification Problem?**
  - **Follow the Steps mentioned in this Lecture and**
    - **How will you Develop a Heart Disease Classification System using Train-Test Split Approach?**

SLIDE
Your Turn

- **Task**
  - **Select a Problem (similar to the one given in TODO) and answer the questions given below**
- **Note**
  - **Your answer should be**
    - **Well Justified**
- **Questions**
  - **Write Input and Output for the selected Machine Learning Problem?**
  - **Follow the Steps mentioned in this Lecture and**

- **How you will Develop a Classification System for the selected Machine Learning Problem using Train-Test Split Approach?**

Lecture Summary

SLIDE

Lecture Summary

- Titanic Passenger Survival Prediction System – Task
  - Given
    - A Passenger (Represented as Set of Attributes)
  - Task
    - Automatically predict whether the Passenger Survived or Not
- Titanic Passenger Survival Prediction System – Input and Output
  - Input
    - A Passenger
  - Output
    - Survived / Not Survived
- The Problem of Titanic Passenger Survival Prediction is treated as a
  - Supervised Machine Learning Task
- The main goal of Titanic Passenger Survival Prediction System is to
  - Learn an Input-Output Function
    - i.e. Learn from Input to predict the Output
- Learning Input-Output Function – General Settings
  - 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)
- The Four main components of an Experimental Setup are
  - Dataset
  - Technique

- Evaluation Methodology
- Evaluation Measures

- In Coding Setup you should clearly write
  - System Settings
  - Libraries
  - Built-in Functions
  - User-Defined Functions
  - Basic Terms
  - Variable Names
- For any 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