**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)

**Input:**

* 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/code/cdabakoglu/heart-disease-classifications-machine-learning/notebook**](https://www.kaggle.com/code/cdabakoglu/heart-disease-classifications-machine-learning/notebook)
* **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**
* **Attribute 01 – Age:**
  + age in years
* **Attribute 02 – Sex:**
  + 1 = male
  + 0 = female
* **Attribute 03 – Cp:**
  + Possible Value 01 = Zero
  + Possible Value 02 = One
  + Possible Value 03 = Two
  + Possible Value 04 = Three
* **Attribute 04 – Chol:**
  + chol - serum cholestoral in mg/dl

**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**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute no.** | **Attribute Names** | **Possible Values** | **Data Types** |
| **1** | **Age** | **Age in years** | **Regression** |
| **2** | **Sex** | **Male(1), Female(0)** | **Categorical** |
| **3** | **CP** | **Zero, One, Two, Three** | **Categorical** |
| **4** | **Col** | **Cholesterol Measures** | **Regression** |
| **5** | **Target** | **Yes, No** | **Categorical** |

**SLIDE:**

**Heart-Disease Prediction System:**

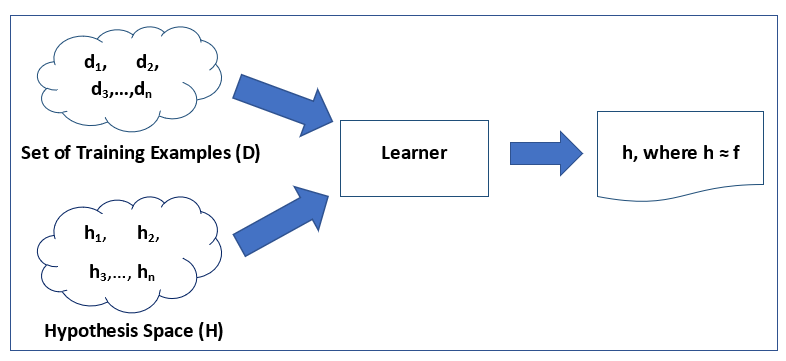
* **Task**
  + **Develop a Heart-Disease Prediction System to Predict the Disease of a Patient.**
* **Input**
  + **Four Attributes**

|  |
| --- |
| 1. **Age** 2. **Sex** 3. **CP** 4. **Col** |

* **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**

* **The Dataset (or Sample Data), used for this Lecture comprises of** 
  + **100 Instances**
    - **See heart-disease-sample-data.csv File in the Data and Code**
* **Sample Data Characteristics** 
  + **Total Instances in Sample Data = 100**
    - **Have Disease = 50**
    - **Not Have Disease = 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 😊**

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

**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**

**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 Heart Disease 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….**

* **To Train / Test Support Vector Classifier, In Sha Allah, we will use**
  + **K-Fold Cross Validation Approach**
* **Question**
  + **How main Folds will we have considering our Sample Data of 100 instances?**
* **Answer**
  + **Each Fold must have at least 30 instances**
    - **Value of K = 100 / 30 = 3.33**
    - **Value of K = 3**
  + **We will apply 3-Fold Cross Validation**
* **Splitting Data into 3-Folds**
  + **Fold 01 = 1 – 34 (total 34 instances)**
  + **Fold 02 = 35 – 67 (total 33 instances)**
* **Fold 03 = 68 – 100 (total 33 instances)**
* **The following Table shows how we will apply 3-Fold Cross-Validation Approach to Train / Test Support Vector Classifier**

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration No.** | **Fold 1** | **Fold 2** | **Fold 3** |
| **Iteration # 1** | Test |  |  |
| **Iteration # 2** |  | Test |  |
| **Iteration # 3** |  |  | Test |

|  |
| --- |
| **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**

|  |
| --- |
|  |

* **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. Muhammad Taimoor** |
| **Programming Language** | **Python 3.8.4** |
| **IDE** | **Jupyter Notebook 6.4.1** |
| **Machine Learning Toolkit** | **Scikit Learn 0.21.2** |
| **Code Version** | **1.0** |
| **Date** | **07 – April – 2022** |

|  |
| --- |
| **Libraries** |

**Libraries**

* **In this Lecture, I used the following Libraries to Write Code for**
  + **Developing a Titanic Passenger Survival Prediction System using K-Fold Cross Validation 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 learning methods used for**[**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 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** | [**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/) |

|  |  |
| --- | --- |
| **KFold** | |
| **Definition** | **K-Folds cross-validator.** |
| **Purpose** | **Provides train/test indices to split data into train/test sets. Split dataset into k consecutive folds.** |
| **Documentation Link** | **https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.KFold.html** |

|  |  |
| --- | --- |
| **Accuracy\_Score** | |
| **Definition** | **Accuracy is defined as the proportion of correctly classified Test Instances.** |
| **Purpose** | **Calculate Accuracy Score.** |
| **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) |

**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 K-Fold Cross Validation 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 K-Fold Cross Validation 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** | **KFold()** |
| **Purpose** | **Provides train/test indices to split data into train/test sets** |

|  |  |
| --- | --- |
| **Function 15** | |
| **Function Name** | **np.ravel()** |
| **Purpose** | **Used to create a contiguous Flattened Array** |

**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 K-Fold Cross Validation Approach**

|  |
| --- |
| **User-Defined Functions** |

**SLIDE**

**User-Defined Functions**

* **In this Lecture, I used the following User Defined Functions to Write Code for**
  + **Developing a Titanic Passenger Survival Prediction System using K-Fold Cross Validation Approach**

|  |  |
| --- | --- |
| **Function 01** | |
| **Function Name** | **save()** |
| **Purpose** | **Save all Trained Models** |

**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 K-Fold Cross Validation 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 K-Fold Cross Validation Approach**

|  |  |
| --- | --- |
| **Basic Terms** | |
| **Sample Data** | **Numerical Input Values** |
| **Training Data** | **Numerical Output Values** |
| **Testing Data** | **Output Labels** |
| **Survived** | **Machine Learning Algorithms** |
| **Not Survived** | **Input Vectors** |
| **Training Data Encoded** | **Predictions** |
| **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 K-Fold Cross Validation 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 K-Fold Cross Validation Approach**

|  |  |
| --- | --- |
| **Variable Names** | |
| **sample\_data** | **training\_data** |
| **sample\_data\_encoded\_output** | **testing\_data** |
| **sample\_data\_encoded** | **input\_training\_data** |
| **age** | **output\_training\_data** |
| **sex** | **input\_testing\_data** |
| **cp** | **output\_testing\_data** |
| **chol** | **no\_of\_folds** |
| **target** | **svc\_model** |
| **age\_label\_encoder** | **svc\_trained\_model** |
| **sex\_label\_encoder** | **accuracy** |
| **cp\_label\_encoder** | **accuracy\_average** |
| **chol\_label\_encoder** | **accuracy\_list** |
| **input\_vector\_sample\_data** | **model\_predications** |
| **output\_label\_sample\_data** | **user\_input** |
| **age\_input** | **sibling\_input** |
| **sex\_input** | **embarked\_input** |
| **unseen\_data\_features** | **model** |
| **predicted\_target** | **pretty\_table** |

|  |
| --- |
| **Heart Disease Prediction System using K-Fold Cross-Validation 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 Heart Disease Prediction System using K-Fold Cross-Validation Approach.ipynb File in Data and Code**
* **Note** 
  + **Below Code does not contain Output**
  + **In Heart Disease Prediction System using K-Fold Cross-Validation 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 Input Vectors and Outputs/Labels of Sample Data**
    - **Step 6.2: Splitting Sample Data using K-Fold Cross-Validation Approach ( K=3 )**
    - **Step 6.3: Train the Support Vector Classifier**
    - **Step 6.4: Save the Trained Models**
  + **Step 7: Execute the Testing Phase** 
    - **Step 7.1: Load the Saved Models**
    - **Step 7.2: Evaluate the Machine Learning Models**
      * **Step 7.2.1: Make Predictions with the Models on Test Data**
    - **Step 7.3: Calculate the Average 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 Best Model**
    - **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**
  + **Step 9: Execute the Feedback Phase**
  + **Step 10: Improve the Model based on Feedback**

**CODE:**

# Import Libraries

import numpy as np

import pandas as pd

import pickle

from sklearn.model\_selection import KFold

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("F:\Comsats Files\Sem 6\Machine Learning\Assignment 2\Heart Disease\K-Fold\Lecture 2\heart-disease-sample-data-encoding.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")

# Labels

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

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

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

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

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

# 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))

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

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

# Print Original and Encoded Ouput Sample Data

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

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\nage 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"] = chol\_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[['encoded\_age', 'encoded\_sex', 'encoded\_cp', 'encoded\_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)

print("\n\nInput Vectors (Feature Vectors) of Sample Data:")

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

input\_vector\_sample\_data = sample\_data\_encoded.iloc[: , :-1]

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

print(input\_vector\_sample\_data)

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

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

output\_label\_sample\_data = sample\_data\_encoded.iloc[: ,-1]

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

print(" Target")

print(output\_label\_sample\_data)

# Save the Input Vector and Output-Label into CSV File

input\_vector\_sample\_data.to\_csv(r'input-vector-sample-data.csv', index = False, header = True)

output\_label\_sample\_data.to\_csv(r'output-label-sample-data.csv', index = False, header = True)

cv = KFold(n\_splits=3, random\_state=0, shuffle=True)

training\_data = {};

testing\_data = {};

input\_training\_data = {};

output\_training\_data = {};

input\_testing\_data = {};

output\_testing\_data = {};

no\_of\_folds = 0;

for train\_index, test\_index in cv.split(input\_vector\_sample\_data):

# Training Data

training\_data[no\_of\_folds]=sample\_data\_encoded.iloc[train\_index]

input\_training\_data[no\_of\_folds]=input\_vector\_sample\_data.iloc[train\_index]

output\_training\_data[no\_of\_folds]=output\_label\_sample\_data.iloc[train\_index]

# Testing Data

testing\_data[no\_of\_folds]=sample\_data\_encoded.iloc[test\_index]

input\_testing\_data[no\_of\_folds]=input\_vector\_sample\_data.iloc[test\_index]

output\_testing\_data[no\_of\_folds]=output\_label\_sample\_data.iloc[test\_index]

no\_of\_folds += 1

# Save To CSV Files

training\_data[no\_of\_folds-1].to\_csv(r'training-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

input\_training\_data[no\_of\_folds-1].to\_csv(r'input-training-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

output\_training\_data[no\_of\_folds-1].to\_csv(r'output-training-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

testing\_data[no\_of\_folds-1].to\_csv(r'testing-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

input\_testing\_data[no\_of\_folds-1].to\_csv(r'input-testing-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

output\_testing\_data[no\_of\_folds-1].to\_csv(r'output-testing-data-iteration-0'+str(no\_of\_folds)+'.csv', index = False, header = True)

for i in range(no\_of\_folds):

# Print Training Data of Each Iteration

print("\n\nTraining Data Input Vectors (Feature Vectots) for Iteration 0" + str(i+1) + " :")

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

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

print("\n",input\_training\_data[i])

print("\n\nTraining Data Outputs/Labels for Iteration 0" + str(i+1) + " :")

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

print(" Survived")

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

print(output\_training\_data[i])

# Train the Models

print("\n\nTraining the Support Vector Classifier for Iteration 0" + str(i+1) + " :")

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

print("Parameters and their values:")

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

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

svc\_model.fit(input\_training\_data[i],output\_training\_data[i])

print(svc\_model)

# Save the Models in a Pkl File

def save(svc\_model):

pickle.dump(svc\_model, open('svc\_trained\_model\_0'+str(i+1)+'.pkl', 'wb'))

svc\_trained\_model = {}

# Load the Saved Models

for i in range(no\_of\_folds):

svc\_trained\_model[i] = pickle.load(open('svc\_trained\_model\_0'+str(i+1)+'.pkl', 'rb'))

# Provide Test data to the Trained Models

accuracy\_list = []

for i in range(no\_of\_folds):

print("\n\nTesting Phase for Iteration 0" + str(i+1) + " :")

print("================================")

print("\nPredictions returned by svc\_trained\_model 0" + str(i+1) + " :")

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

model\_predications = svc\_trained\_model[i].predict(input\_testing\_data[i])

model\_predications\_data = input\_testing\_data[i].copy()

model\_predications\_data["Target"] = output\_testing\_data[i]

model\_predications\_data["Predictions"] = model\_predications

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

print(model\_predications\_data)

# Save the Predictions into CSV File

model\_predications\_data.to\_csv(r'model-predictions-iteration-0' + str(i+1) + '.csv', index = False, header = True)

# Calculate the Accuracy of each Iteration

print("\n\nAccuracy Score:")

print("===============")

accuracy = accuracy\_score(model\_predications\_data["Target"],model\_predications\_data["Predictions"])

accuracy\_list.append(accuracy);

print(round(accuracy,2))

# Calculate the Average Accuracy

print("\n\nAverage Accuracy Score:")

print("=======================")

accuracy\_average = sum(accuracy\_list) / len(accuracy\_list)

print(round(accuracy\_average,2))

# Take Input from User

age\_input = input("\nPlease enter age here (old,mature) : ").strip()

sex\_input = input("\nPlease enter your Gender here (male,female) : ").strip()

cp\_input = input("\nPlease enter your Sibling here (Zero,One,Two,Three) : ").strip()

chol\_input = input("\nPlease enter Embarked here (zero,one) : ").strip()

# Convert User Input into Feature Vector

user\_input = pd.DataFrame({ 'age': [age\_input],'sex': [sex\_input],'cp': [cp\_input],'chol': [chol\_input]})

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

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

print(user\_input)

# Transform Input (Categorical) Attributes of Unseen Data into Numerical Representation

unseen\_data\_features = user\_input.copy()

unseen\_data\_features["age"] = age\_label\_encoder.transform(user\_input['age'])

unseen\_data\_features["sex"] = sex\_label\_encoder.transform(user\_input['sex'])

unseen\_data\_features["cp"] = cp\_label\_encoder.transform(user\_input['cp'])

unseen\_data\_features["chol"] = chol\_label\_encoder.transform(user\_input['chol'])

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

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

print(user\_input)

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

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

print(unseen\_data\_features)

# Load the Best Model

# svc\_trained\_model\_01 has Highest Accuracy

model = pickle.load(open('svc\_trained\_model\_0.pkl', 'rb'))

# Make a Prediction on Unseen Data

predicted\_survival = model.predict(unseen\_data\_features)

if(predicted\_survival == 1):

prediction = "Have Disease"

if(predicted\_survival == 0):

prediction = "NOT Have Disease"

# Add the Prediction in a Pretty Table

pretty\_table = PrettyTable()

pretty\_table.add\_column(" \*\* Prediction \*\* ",[prediction])

print(pretty\_table)