SUPERIOR UNIVERSITY

Name : MUHAMMAD AHMAD

Roll No : SU92-BSAIM-F23-135

Section : 4-C

Task : 02

Date : 05th March, 2025

Subject : Programming AI Lab

Submitted To : Prof Rasikh Ali

TASK : 02

Titanic Survival Predictor

1. Introduction

This report outlines the steps taken in the PAI Lab Task 02, which involves data preprocessing, feature selection, model training, and evaluation for the given dataset. The dataset appears to be related to space travel passengers.

2. Data Exploration

The dataset is loaded using Pandas, and an initial analysis is conducted to check the structure and missing values.

```
import pandas as pd

train_data = pd.read_csv('train_space.csv')

test_X = pd.read_csv('test_space.csv')

test_y = pd.read_csv('sample_submission_space.csv')


train_data.head()

test_X.head()

test_y.head()


train_data.info()

train_data.isnull().sum()
```

3. Data Preprocessing

Feature Selection

Some columns are removed due to irrelevance or excessive missing values.

```
def drop_col(df, col):

    for i in col:

        df.drop(i, axis=1, inplace=True)
```

```
col = ['PassengerId', 'CryoSleep', 'Cabin', 'VIP', 'Name']

drop_col(encoded_train, col)

drop_col(encoded_test_X, col)

drop_col(encoded_test_y, ['PassengerId'])
```

Handling Missing Values and Outliers

Numerical missing values are handled using the interquartile range (IQR) method to remove outliers, then filling missing values with the mean.

```
def fill_null(df, col):

    for i in col:

        q1 = df[i].quantile(0.25)

        q3 = df[i].quantile(0.75)

        iqr = q3 - q1

        lower_lim = q1 - 1.5 * iqr

        upper_lim = q3 + 1.5 * iqr

        without_outlier = df[i].apply(lambda x: None if x < lower_lim or x > upper_lim else x)

        without_outlier.fillna(without_outlier.mean(), inplace=True)

        df[i] = without_outlier
```

Categorical missing values are filled with the most frequent value (mode).

```
def fill_obj(df, col):

    for i in col:

        df[i].fillna(df[i].mode()[0], inplace=True)
```

4. Data Transformation

Identifying Column Types

```
def int_col(df):

    return [i for i in df.columns if df[i].dtype != 'O']
```

```
def obj_col(df):

    return [i for i in df.columns if df[i].dtype == 'O']
```

Feature Scaling

Standard Scaling is applied to continuous variables.

```
from sklearn.preprocessing import StandardScaler

std = StandardScaler()

std_list = ['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']

encoded_train[std_list] = std.fit_transform(encoded_train[std_list])

encoded_test_X[std_list] = std.transform(encoded_test_X[std_list])
```

MinMax Scaling is applied to categorical numerical features.

```
from sklearn.preprocessing import MinMaxScaler

minmax = MinMaxScaler()

minmax_list = ['Transported']

encoded_train[minmax_list] = minmax.fit_transform(encoded_train[minmax_list])

encoded_test_X[minmax_list] = minmax.transform(encoded_test_X[minmax_list])
```

Encoding Categorical Features

```
from sklearn.preprocessing import LabelEncoder



def obj_to_int(df, col):

    for i in col:

        label = LabelEncoder()

        df[i] = label.fit_transform(df[i])
```

5. Model Training

Splitting features and target variable:

```
train_X = encoded_train.drop('Transported', axis=1)
```

train_y = encoded_train['Transported']

Training a Support Vector Machine model:

from sklearn.svm import SVC

svc_model = SVC()

svc_model.fit(train_X, train_y)

6. Predictions

pred = svc_model.predict(encoded_test_X)

7. Conclusion

This project covers data preprocessing, feature engineering, and training an SVM model for classification. Further improvements can be made by testing other models such as Random Forest or Neural Networks.