**A purple circle with dots and lines

Description automatically generated**

**NAME:**

**Basit Abbas**

**ROLL NUMBER:**

**SU92-BSDSM-F23- 013**

**PROGRAM:**

**BS DATA SCIENCE**

**SUMBITTED TO:**

**SIR RASIKH**

***TASK : 02***

**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.