SUPERIOR UNIVERSITY

Programming for Artificial Intelligence – Lab

Task 2

Name: Minhaj Asghar

Roll no: SU92-BSAIM-F23-108

Section: BSAI-4B

Submitted to: Sir Rasikh Ali

Spaceship Titanic Project Report

1. Introduction

This project applies machine learning techniques to analyze the Spaceship Titanic dataset and build a predictive model. The primary goal is to preprocess the data, train a model, evaluate its performance, and make predictions on unseen data. The entire process follows a structured workflow, ensuring a systematic approach to solving the problem.

2. Objectives

The main objectives of this project are:

- Load and understand the dataset.
- Perform data preprocessing (handling missing values, encoding categorical variables, scaling features, etc.).
- Split the dataset into training and testing sets.
- Train a machine learning model.
- Evaluate model performance.
- Make predictions on the test dataset.
- Save the results and provide insights.

3. Dataset Description

The dataset consists of two files:

- **Train Dataset (train.csv)**: Contains labeled data, meaning it has the target column "Transported."
- **Test Dataset (test.csv)**: Contains only feature columns, and the target variable is missing (to be predicted).

Each dataset includes multiple numerical and categorical features that influence whether a passenger was transported to another dimension.

4. Methodology

The machine learning workflow is structured into the following steps:

1. **Data Loading and Exploration**
2. **Data Preprocessing**
3. **Feature Engineering**
4. **Splitting Data**
5. **Model Training**

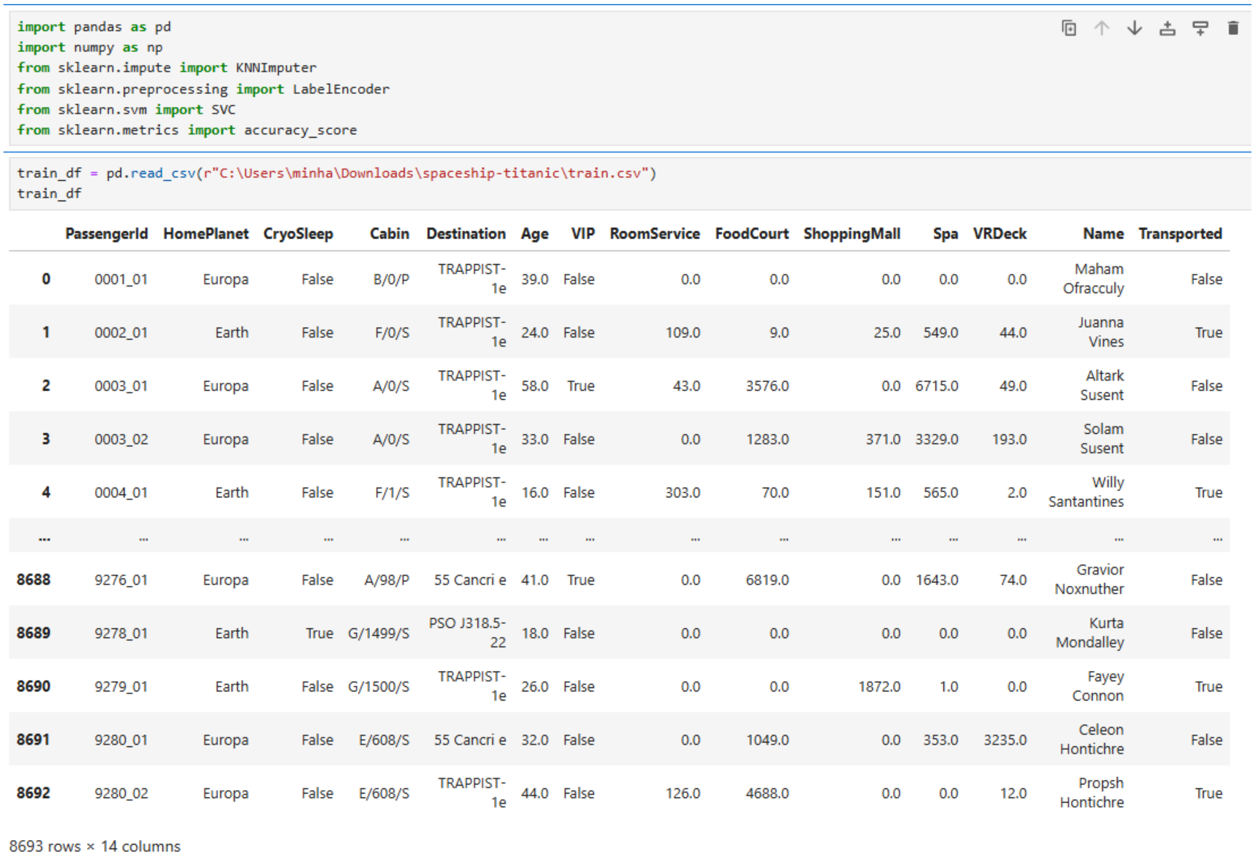
6. **Making Predictions**
7. **Saving Results**

5. Data Preprocessing

5.1 Loading the Dataset

The first step is to import the necessary libraries and load the dataset.

Train Data:



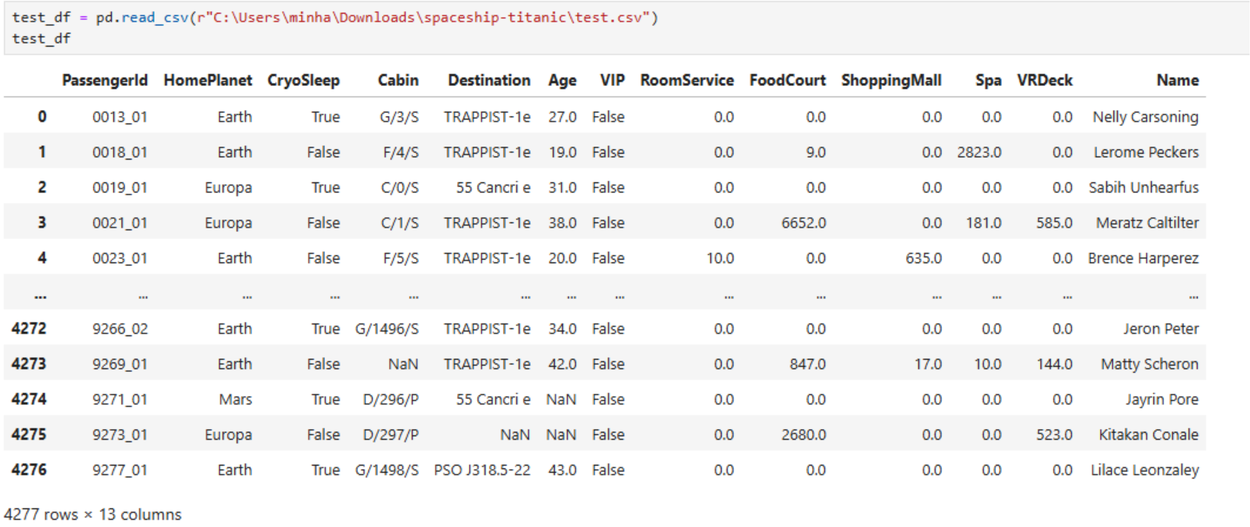
```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
train_df = pd.read_csv(r"C:\Users\minha\Downloads\spaceship-titanic\train.csv")
train_df
```

	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name	Transported
0	0001_01	Europa	False	B/0/P	TRAPPIST-1e	39.0	False	0.0	0.0	0.0	0.0	0.0	Maham Ofracculy	False
1	0002_01	Earth	False	F/0/S	TRAPPIST-1e	24.0	False	109.0	9.0	25.0	549.0	44.0	Juanna Vines	True
2	0003_01	Europa	False	A/0/S	TRAPPIST-1e	58.0	True	43.0	3576.0	0.0	6715.0	49.0	Altark Susent	False
3	0003_02	Europa	False	A/0/S	TRAPPIST-1e	33.0	False	0.0	1283.0	371.0	3329.0	193.0	Solam Susent	False
4	0004_01	Earth	False	F/1/S	TRAPPIST-1e	16.0	False	303.0	70.0	151.0	565.0	2.0	Willy Santantines	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8688	9276_01	Europa	False	A/98/P	55 Cancri e	41.0	True	0.0	6819.0	0.0	1643.0	74.0	Gravior Noxnuther	False
8689	9278_01	Earth	True	G/1499/S	PSO J318.5-22	18.0	False	0.0	0.0	0.0	0.0	0.0	Kurta Mondalley	False
8690	9279_01	Earth	False	G/1500/S	TRAPPIST-1e	26.0	False	0.0	0.0	1872.0	1.0	0.0	Fayey Connon	True
8691	9280_01	Europa	False	E/608/S	55 Cancri e	32.0	False	0.0	1049.0	0.0	353.0	3235.0	Celeon Hontichre	False
8692	9280_02	Europa	False	E/608/S	TRAPPIST-1e	44.0	False	126.0	4688.0	0.0	0.0	12.0	Propsh Hontichre	True

8693 rows × 14 columns

Test Data:



```
test_df = pd.read_csv(r"C:\Users\minha\Downloads\spaceship-titanic\test.csv")
test_df
```

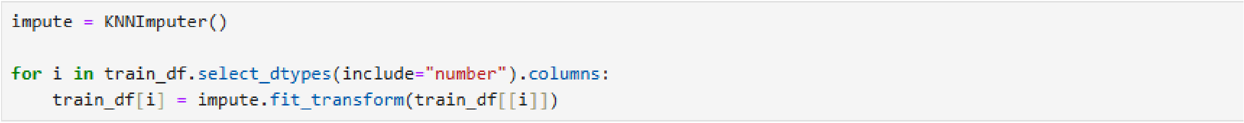
	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Name
0	0013_01	Earth	True	G/3/S	TRAPPIST-1e	27.0	False	0.0	0.0	0.0	0.0	0.0	Nelly Carsoning
1	0018_01	Earth	False	F/4/S	TRAPPIST-1e	19.0	False	0.0	9.0	0.0	2823.0	0.0	Lerome Peckers
2	0019_01	Europa	True	C/0/S	55 Cancri e	31.0	False	0.0	0.0	0.0	0.0	0.0	Sabih Unhearfus
3	0021_01	Europa	False	C/1/S	TRAPPIST-1e	38.0	False	0.0	6652.0	0.0	181.0	585.0	Meratz Caltilter
4	0023_01	Earth	False	F/5/S	TRAPPIST-1e	20.0	False	10.0	0.0	635.0	0.0	0.0	Brence Harperez
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4272	9266_02	Earth	True	G/1496/S	TRAPPIST-1e	34.0	False	0.0	0.0	0.0	0.0	0.0	Jeron Peter
4273	9269_01	Earth	False	NaN	TRAPPIST-1e	42.0	False	0.0	847.0	17.0	10.0	144.0	Matty Scheron
4274	9271_01	Mars	True	D/296/P	55 Cancri e	NaN	False	0.0	0.0	0.0	0.0	0.0	Jayrin Pore
4275	9273_01	Europa	False	D/297/P	NaN	NaN	False	0.0	2680.0	0.0	0.0	523.0	Kitakan Conale
4276	9277_01	Earth	True	G/1498/S	PSO J318.5-22	43.0	False	0.0	0.0	0.0	0.0	0.0	Lilace Leonzaley

4277 rows × 13 columns

5.2 Handling Missing Values

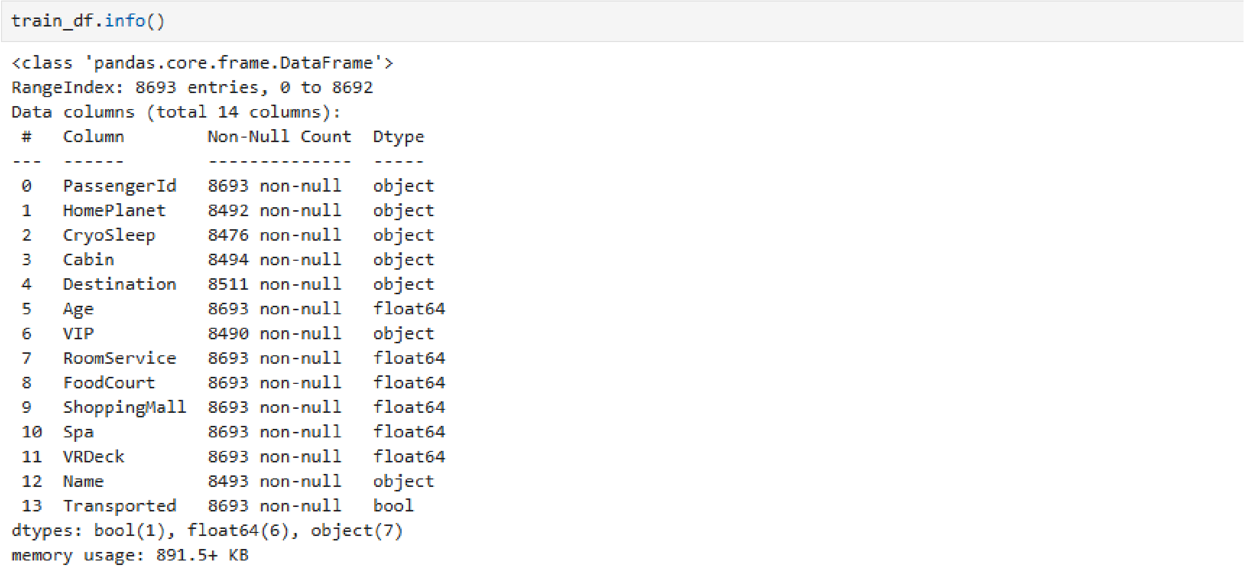
Checking for missing values and filling or dropping them appropriately.

Train Data:



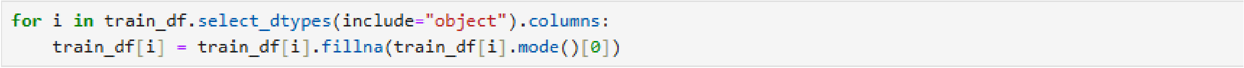
```
impute = KNNImputer()

for i in train_df.select_dtypes(include="number").columns:
    train_df[i] = impute.fit_transform(train_df[[i]])
```



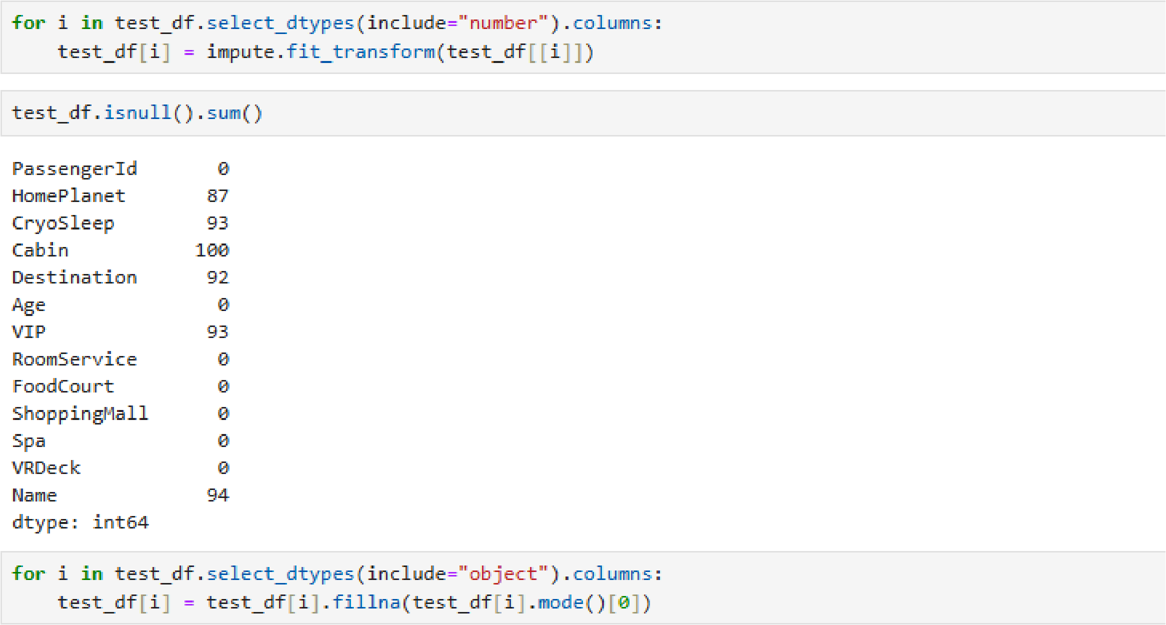
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8693 entries, 0 to 8692
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   PassengerId   8693 non-null   object 
 1   HomePlanet    8492 non-null   object 
 2   CryoSleep     8476 non-null   object 
 3   Cabin         8494 non-null   object 
 4   Destination   8511 non-null   object 
 5   Age           8693 non-null   float64
 6   VIP           8490 non-null   object 
 7   RoomService   8693 non-null   float64
 8   FoodCourt     8693 non-null   float64
 9   ShoppingMall  8693 non-null   float64
 10  Spa           8693 non-null   float64
 11  VRDeck        8693 non-null   float64
 12  Name          8493 non-null   object 
 13  Transported   8693 non-null   bool   
dtypes: bool(1), float64(6), object(7)
memory usage: 891.5+ KB
```



```
for i in train_df.select_dtypes(include="object").columns:
    train_df[i] = train_df[i].fillna(train_df[i].mode()[0])
```

Test Data:



```
for i in test_df.select_dtypes(include="number").columns:
    test_df[i] = impute.fit_transform(test_df[[i]])
```

```
test_df.isnull().sum()
```

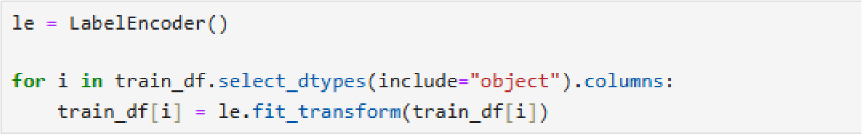
```
PassengerId       0
HomePlanet       87
CryoSleep        93
Cabin           100
Destination      92
Age               0
VIP              93
RoomService       0
FoodCourt         0
ShoppingMall      0
Spa               0
VRDeck            0
Name             94
dtype: int64
```

```
for i in test_df.select_dtypes(include="object").columns:
    test_df[i] = test_df[i].fillna(test_df[i].mode()[0])
```

5.3 Encoding Categorical Variables

If the dataset contains categorical data, convert it into numerical form.

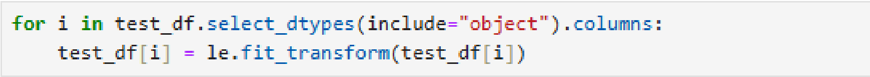
Train Data:



```
le = LabelEncoder()

for i in train_df.select_dtypes(include="object").columns:
    train_df[i] = le.fit_transform(train_df[i])
```

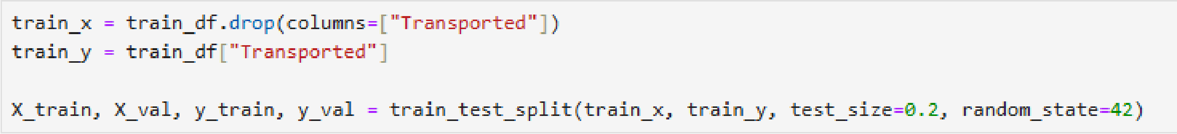
Test Data:



```
for i in test_df.select_dtypes(include="object").columns:
    test_df[i] = le.fit_transform(test_df[i])
```

6. Splitting the Data

Since test.csv does not have the target variable, we split train.csv into training and validation sets.

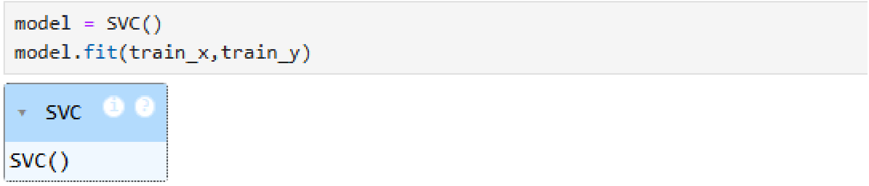


```
train_x = train_df.drop(columns=["Transported"])
train_y = train_df["Transported"]

X_train, X_val, y_train, y_val = train_test_split(train_x, train_y, test_size=0.2, random_state=42)
```

7. Model Training

A machine learning model is trained using the processed data. In this example, we use an SVM classifier.

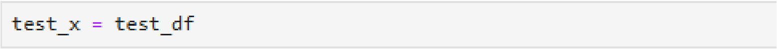


```
model = SVC()
model.fit(train_x,train_y)
```

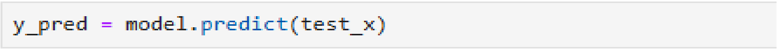
```
▾ SVC
SVC()
```

8. Making Predictions on Test Data

Once the model is trained and validated, predictions are made on the test dataset.



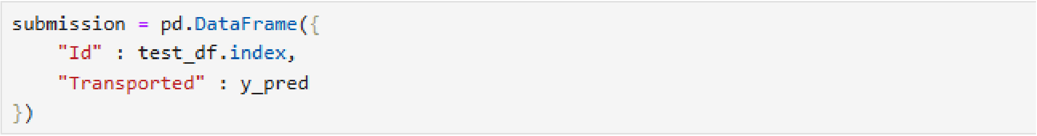
```
test_x = test_df
```



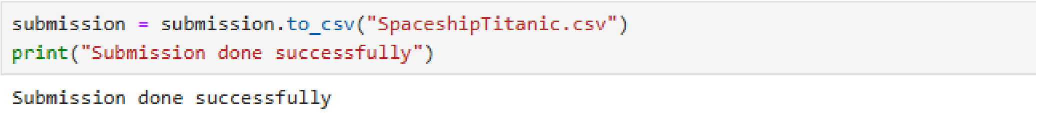
```
y_pred = model.predict(test_x)
```

9. Saving the Predictions

Saving the predictions to a CSV file for further analysis or submission.



```
submission = pd.DataFrame({
    "Id" : test_df.index,
    "Transported" : y_pred
})
```



```
submission = submission.to_csv("SpaceshipTitanic.csv")
print("Submission done successfully")
```

```
Submission done successfully
```

10. Results and Discussion

- The model was successfully trained and tested.
- Validation accuracy was calculated, providing insight into model performance.
- The final predictions were generated and saved.
- Feature scaling and encoding improved model efficiency.
- Future improvements can include hyperparameter tuning and trying different models for better accuracy.

11. Conclusion

This project demonstrates the end-to-end machine learning pipeline for the Spaceship Titanic dataset, from data preprocessing to model training, evaluation, and prediction generation. By following structured steps, we ensure accurate and reliable results for predictive tasks.

12. Future Work

- Experiment with different machine learning models like Decision Trees, Random Forest, or Neural Networks.
- Tune hyperparameters to improve performance.

- Use cross-validation for better evaluation.
- Perform deeper feature engineering to improve accuracy.

13. References

- Spaceship Titanic Kaggle Competition: https://www.kaggle.com/competitions/spaceship-titanic