Machine Learning Lab - 2

DATA PREPROCESSING

To apply various machine learning techniques to answer different questions, you should start by cleaning and preprocessing the data for each classification problem. For instance, with the Titanic dataset, you can prepare the data to be suitable for training and evaluating models for predicting or inferring demographic characteristics. Use this dataset to address eight distinct classification problems given below, ensuring that the data is properly preprocessed and that suitable features are selected for each model.

Instructions:

1. Data Preprocessing:
   - Load the Titanic dataset and clean basic data.
   - Handle missing values appropriately.
   - Convert categorical variables into numerical representations.

Output -

```
# Importing Dataset
import pandas as pd
data = pd.read_excel("/content/drive/MyDrive/Lab/ML/Lab_2/titanic.xlsx")
data.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	0	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	1	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	0	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	1	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	0	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

```
missing_data = data.isna().sum()
missing_data
```

Udit Jain
24CSM1R23

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

```
# Age Null Values Handling
mean_age = data["Age"].mean()
mean_age = int(mean_age)

data["Age"] = data["Age"].apply(lambda ele : mean_age if pd.isna(ele) else
ele)
data.head(50)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	0	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	1	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	0	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	1	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C

```
# Fare Null Values Handling
mean_fare = data["Fare"].mean()
```

Udit Jain
24CSM1R23

```
mean_fare = int(mean_fare)

data["Fare"] = data["Fare"].apply(lambda ele : mean_fare if pd.isna(ele)
else ele)
data.head(20)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	0	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	1	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	0	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	1	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C

```
# HasCabin Label Encoding
data["HasCabin"] = data["Cabin"].apply(lambda ele : 0 if pd.isna(ele) else
1)
```

```
# Sex Label Encoding
# male -> 0
# female -> 1
data["Sex"] = data["Sex"].map(lambda ele : 0 if ele == 'male' else 1)
```

```
# Embarked Label Encoding
# C -> 0
# Q -> 1
# s -> 2
data["Embarked"] = data["Embarked"].map({ 'C' : 0, 'Q' : 1, 'S' : 2 })
```

```
# Family Status Label Encoding
data["FamilyStatus"] = data.apply(lambda row : 1 if row["SibSp"] > 0 or
row["Parch"] > 0 else 0, axis = 1)
```

```
# HighFare Label Encoding
data["HighFare"] = data.apply(lambda row : 1 if row["Fare"] > mean_fare
else 0, axis = 1)
```

```
# Age Group Label Encoding
max_age = data["Age"].max() # 76.0
```

Udit Jain
24CSM1R23

```
min_age = data["Age"].min() # 0.17

bins = [0, 10, 18, 40, 70, 100]
labels = [0, 1, 2, 3, 4]
data['AgeGroup'] = pd.cut(data['Age'], bins=bins, labels=labels)
```

```
# Name Label Encoding
uniqueNames = data["Name"].unique()
nameNumberMap = {}
for i, name in enumerate(uniqueNames):
  nameNumberMap[name] = i

data["Name"] = data["Name"].map(nameNumberMap)
```

```
# Ticket Label Encoding
tickets = data["Ticket"]
encodedTickets = []
for ticket in tickets:
  encodedTickets.append(str(ticket)[-5:])

data["Ticket"] = encodedTickets;
```

```
# Cabin Label Encoding
data["Cabin"] = data["Cabin"].map(lambda ele : "Unknown" if pd.isna(ele)
else ele)

data.head(15)

uniqueCabins = data["Cabin"].unique()
cabinNumberMap = {}
for i, name in enumerate(uniqueCabins):
  cabinNumberMap[name] = i

data["Cabin"] = data["Cabin"].map(cabinNumberMap)
```

Udit Jain
24CSM1R23

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	HasCabin	FamilyStatus	HighFare	AgeGroup
0	892	0	3	0	0	34.5	0	0	30911	7.8292	0	1	0	0	0	2
1	893	1	3	1	1	47.0	1	0	63272	7.0000	0	2	0	1	0	3
2	894	0	2	2	0	62.0	0	0	40276	9.6875	0	1	0	0	0	3
3	895	0	3	3	0	27.0	0	0	15154	8.6625	0	2	0	0	0	2
4	896	1	3	4	1	22.0	1	1	01298	12.2875	0	2	0	1	0	2
5	897	0	3	5	0	14.0	0	0	7538	9.2250	0	2	0	0	0	1
6	898	1	3	6	1	30.0	0	0	30972	7.6292	0	1	0	0	0	2
7	899	0	2	7	0	26.0	1	1	48738	29.0000	0	2	0	1	0	2
8	900	1	3	8	1	18.0	0	0	2657	7.2292	0	0	0	0	0	1
9	901	0	3	9	0	21.0	2	0	48871	24.1500	0	2	0	1	0	2

Udit Jain
24CSM1R23