Group 1 - 7W3

TITANIC - MACHINE LEARNING FROM DISASTER

Selected Topics (IT 426T) – Machine Learning

CONTENT

KEY TOPICS DISCUSSED IN THIS PRESENTATION

1. Import and Load Data
2. Data Exploration
3. Data Cleaning
4. Data Visualization and Analysis
5. Data Modeling & Prediction
6. performance Metrics
7. Process for submission file

STEP 1: Import and Load Data

1.1 Import Libraries

```
# linear algebra
import numpy as np

# data processing
import pandas as pd

# data visualization
import seaborn as sns

# Logistic Regression
from sklearn.linear_model import LogisticRegression

# data split
from sklearn.model_selection import train_test_split

# accuracy score
from sklearn.metrics import accuracy_score

# confusion matrix
from sklearn.metrics import confusion_matrix

# performance metrics
from sklearn.metrics import classification_report
```

1.2 Load Data

```
train_data = pd.read_csv("/kaggle/input/titanic/train.csv")
train_data.head() #show the first 5 rows from the training dataset
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
test_data = pd.read_csv("/kaggle/input/titanic/test.csv")
test_data.head() #show the first 5 rows from the testing dataset
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

STEP 2: Data Exploration

2.2 Show If there is null values

```
#confirm that there is null values
train_data.isnull().values.any()
```

```
True
```

```
test_data.isnull().values.any()
```

```
True
```

2.1 Show data Information

```
#display all columns and their data types
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

STEP 3: Data Cleaning

3.1 Convert column names to lowercase

```
#Converting the columns names to lowercase
train_data.columns = [c.lower() for c in train_data.columns]
test_data.columns = [c.lower() for c in test_data.columns]
```

3.2 Rename some columns

```
#Rename columns for train dataset
train_data.rename(columns={
            "passengerid":"passenger_id",
            "pclass":"passenger_class",
            "sibsp":"sibling_spouse",
            "parch":"parent_children"
        }, inplace=True)

#Rename columns for test dataset
test_data.rename(columns={
            "passengerid":"passenger_id",
            "pclass":"passenger_class",
            "sibsp":"sibling_spouse",
            "parch":"parent_children"
        }, inplace=True)
```

3.3 Fill the missing values (CONT.)

```
#fill age missing values with random numbers computed based on mean and the standard deviation
#and change datatype to int on both datasets

for dataset in [train_data, test_data]:
    mean = dataset["age"].mean()
    std = dataset["age"].std()
    is_null = dataset["age"].isnull().sum()

    # compute random numbers between the mean, std and is_null
    random_age = np.random.randint(mean - std, mean + std, size = is_null)

    # fill NaN values in Age column with random values generated
    age_copy = dataset["age"].copy()
    age_copy[np.isnan(age_copy)] = random_age
    dataset["age"] = age_copy
    dataset["age"] = dataset["age"].astype(int)
```

3.3 Fill the missing values (CONT.)

```
#fill the missing values for embarked in the train dataset
train_data.embarked.fillna(train_data.embarked.mode()[0], inplace = True)
```

```
#fill the missing values for fare in the test dataset
test_data.fare.fillna(test_data.fare.mode()[0], inplace = True)
```

3.4 Convert categorical Data to numerical

```
#convert categrical columns to numerical
train_data['sex'].replace(['female','male'],[0,1],inplace = True)
test_data['sex'].replace(['female','male'],[0,1],inplace = True)
```

```
train_data['embarked'].replace(['C','Q','S'],[1,2,3], inplace = True)
test_data['embarked'].replace(['C','Q','S'],[1,2,3], inplace = True)
```

3.4 Drop Columns

```python
#remove columns (name - ticket - cabin)
train_data.drop(labels = ["cabin", "name","ticket"], axis=1, inplace = True)
test_data.drop(labels = ["cabin", "name","ticket"], axis=1, inplace = True)
```

why drop these columns?
Cabin: has too many missing data
Name: not important
Ticket: Contain the ticket number which is not important

3.5 Check age Range values

```
#check that age values are on propore range
train_data.age.min()
```

```
0
```

```
train_data.age.max()
```

```
80
```

3.6 Show data information after cleaning

```
#show data after cleaning
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   passenger_id     891 non-null    int64  
 1   survived         891 non-null    int64  
 2   passenger_class  891 non-null    int64  
 3   sex              891 non-null    int64  
 4   age              891 non-null    int64  
 5   sibling_spouse   891 non-null    int64  
 6   parent_children  891 non-null    int64  
 7   fare             891 non-null    float64
 8   embarked         891 non-null    int64  
dtypes: float64(1), int64(8)
memory usage: 62.8 KB
```

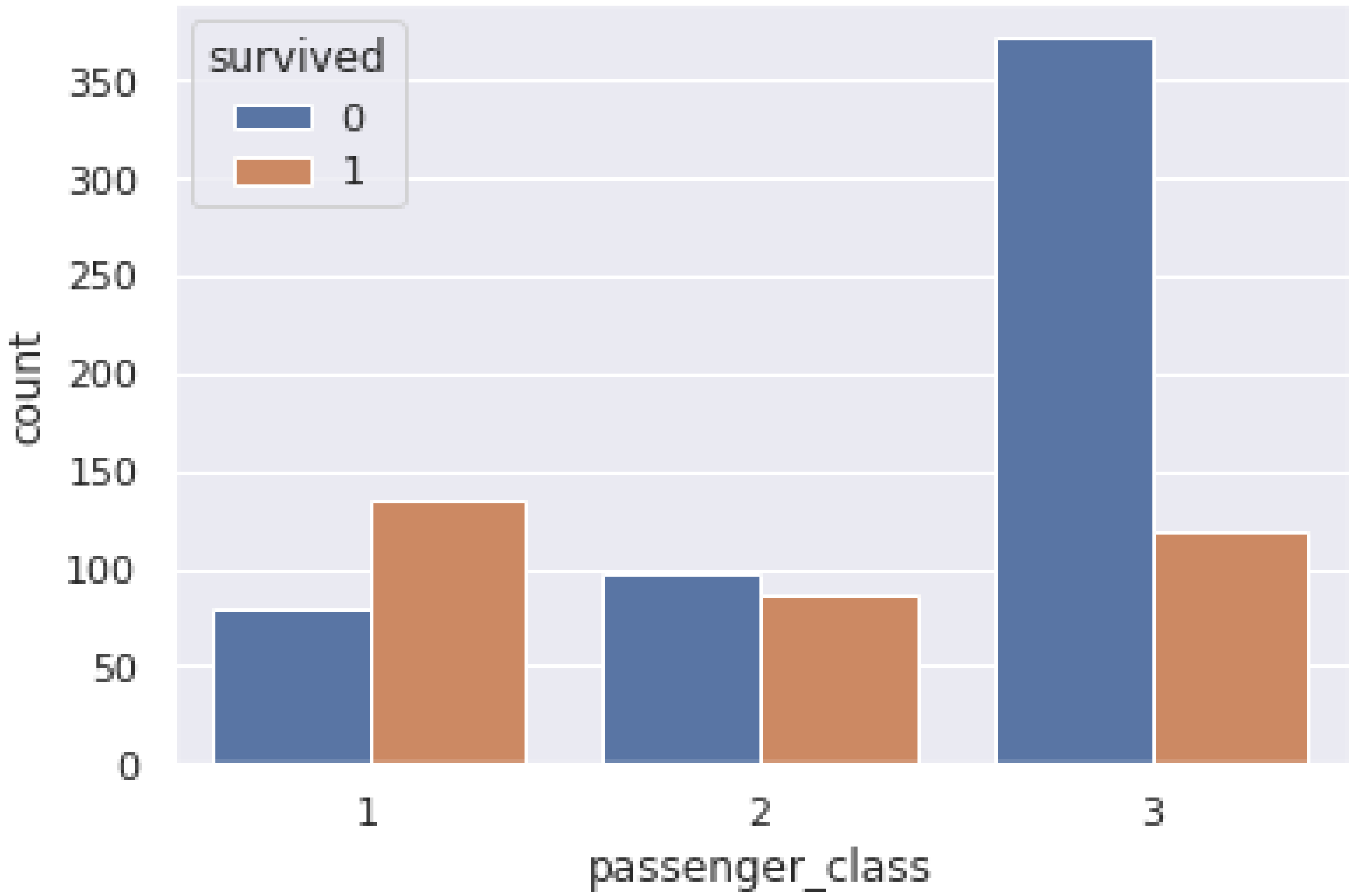
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   passenger_id     418 non-null    int64  
 1   passenger_class  418 non-null    int64  
 2   sex              418 non-null    int64  
 3   age              418 non-null    int64  
 4   sibling_spouse   418 non-null    int64  
 5   parent_children  418 non-null    int64  
 6   fare             418 non-null    float64
 7   embarked         418 non-null    int64  
dtypes: float64(1), int64(7)
memory usage: 26.2 KB
```

STEP 4:
Data Visualization and Analysis

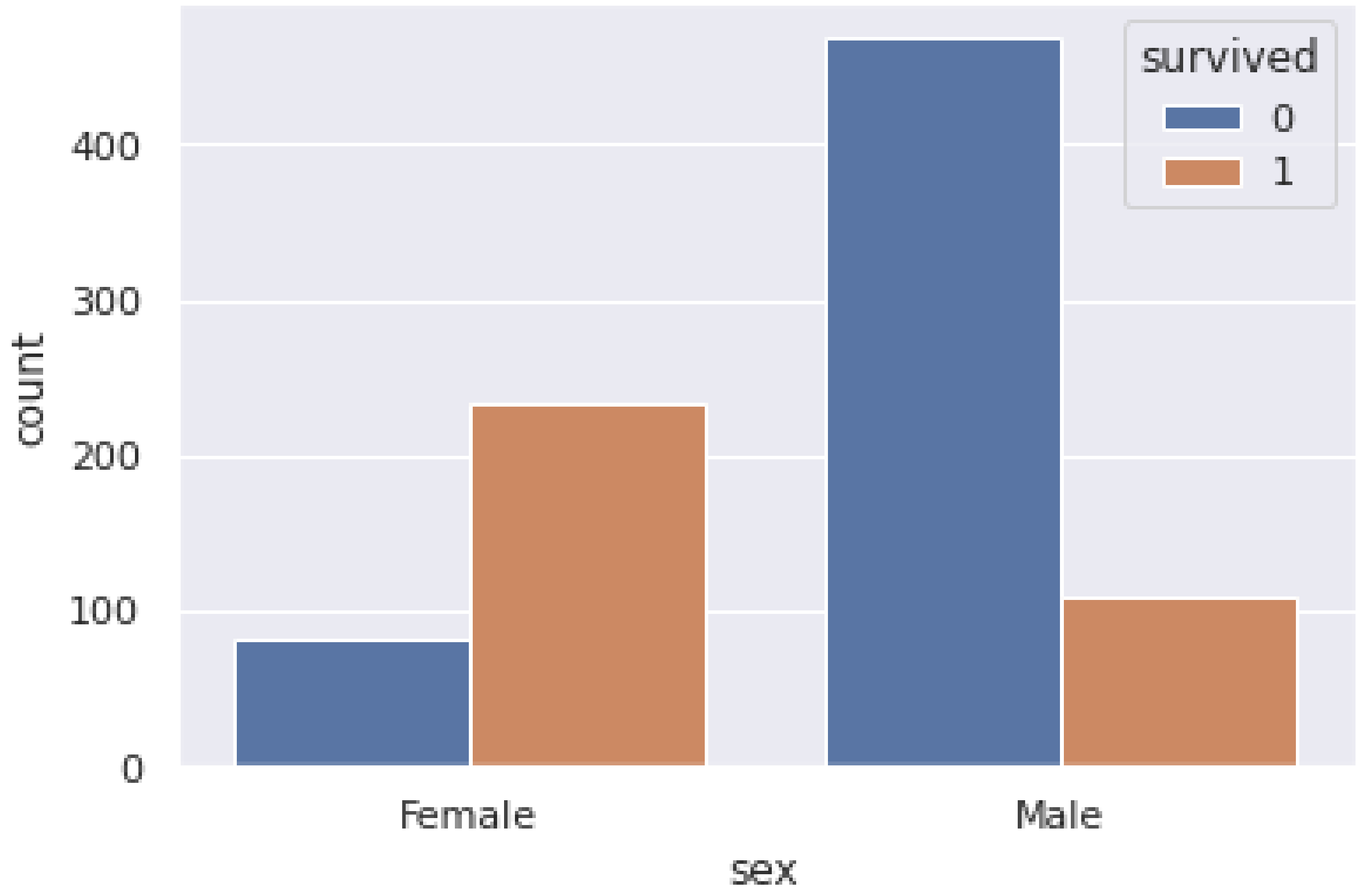
4.1 Did passenger class made any difference to his survival?

```
sns.countplot("passenger_class", data=train_data, hue="survived")
sns.set_theme(style="darkgrid")
```



4.2 Which gender had more survival?

```
data =sns.countplot("sex", data=train_data, hue="survived")
data.set_xticklabels(["Male","Female"])
sns.set_theme(style="darkgrid")
```



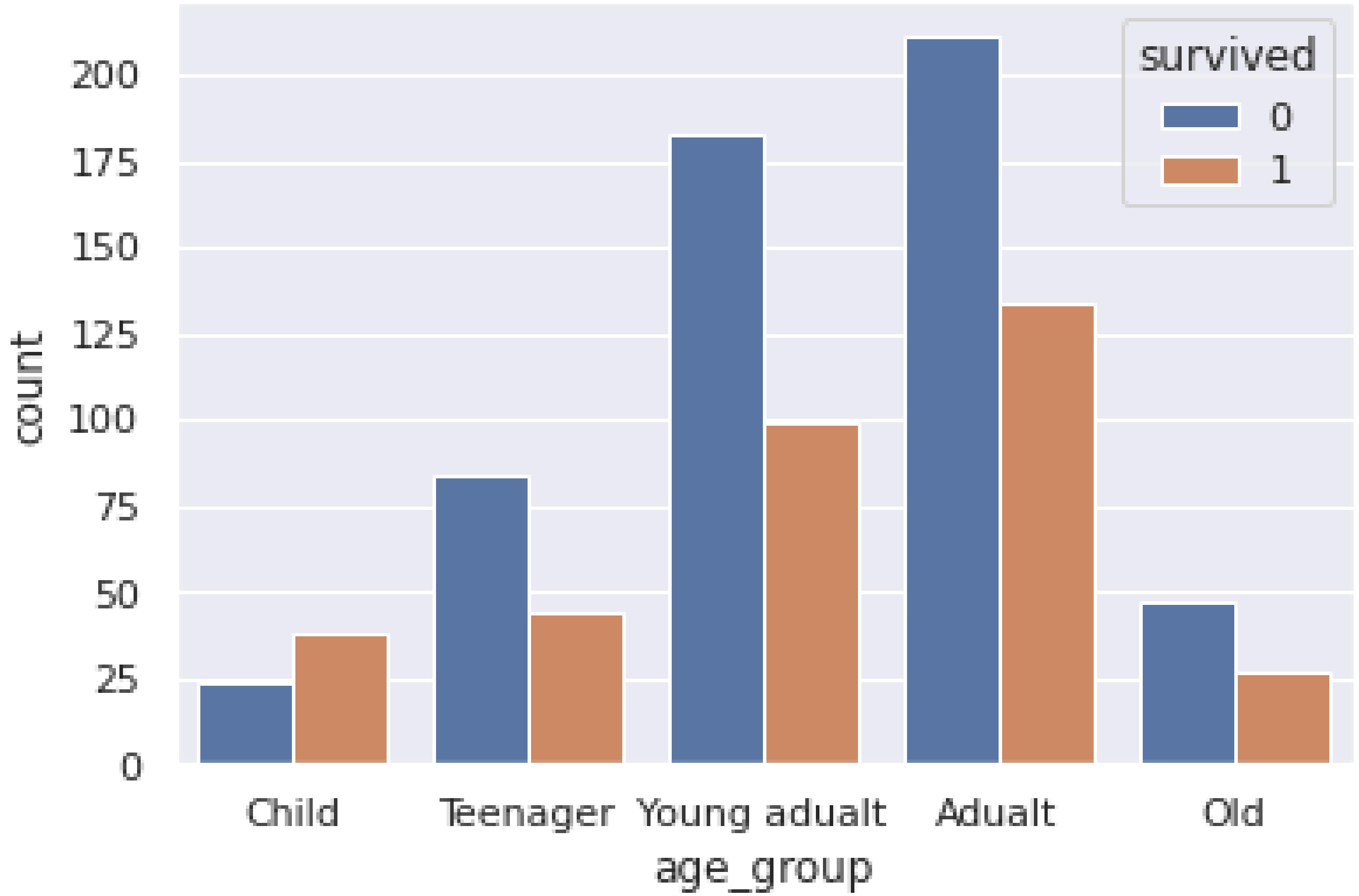
4.3 Which age group had a better chance of survival? (CONT.)

```
 def age_group(age):
  if age >= 50:
      return 'Old'
  if 30 <= age < 50:
      return 'Adualt'
  if  20<= age < 30:
      return 'Young adualt'
  if  10<= age < 20:
      return 'Teenager'
  if  0<= age < 10:
      return 'Child'

train_data['age_group'] =train_data.age.apply(age_group)
```

```
data =sns.countplot("age_group", data=train_data,order=['Child','Teenager','Young adualt','Adualt','Old'], hue="survived")
```

4.3 Which age group had a better chance of survival?



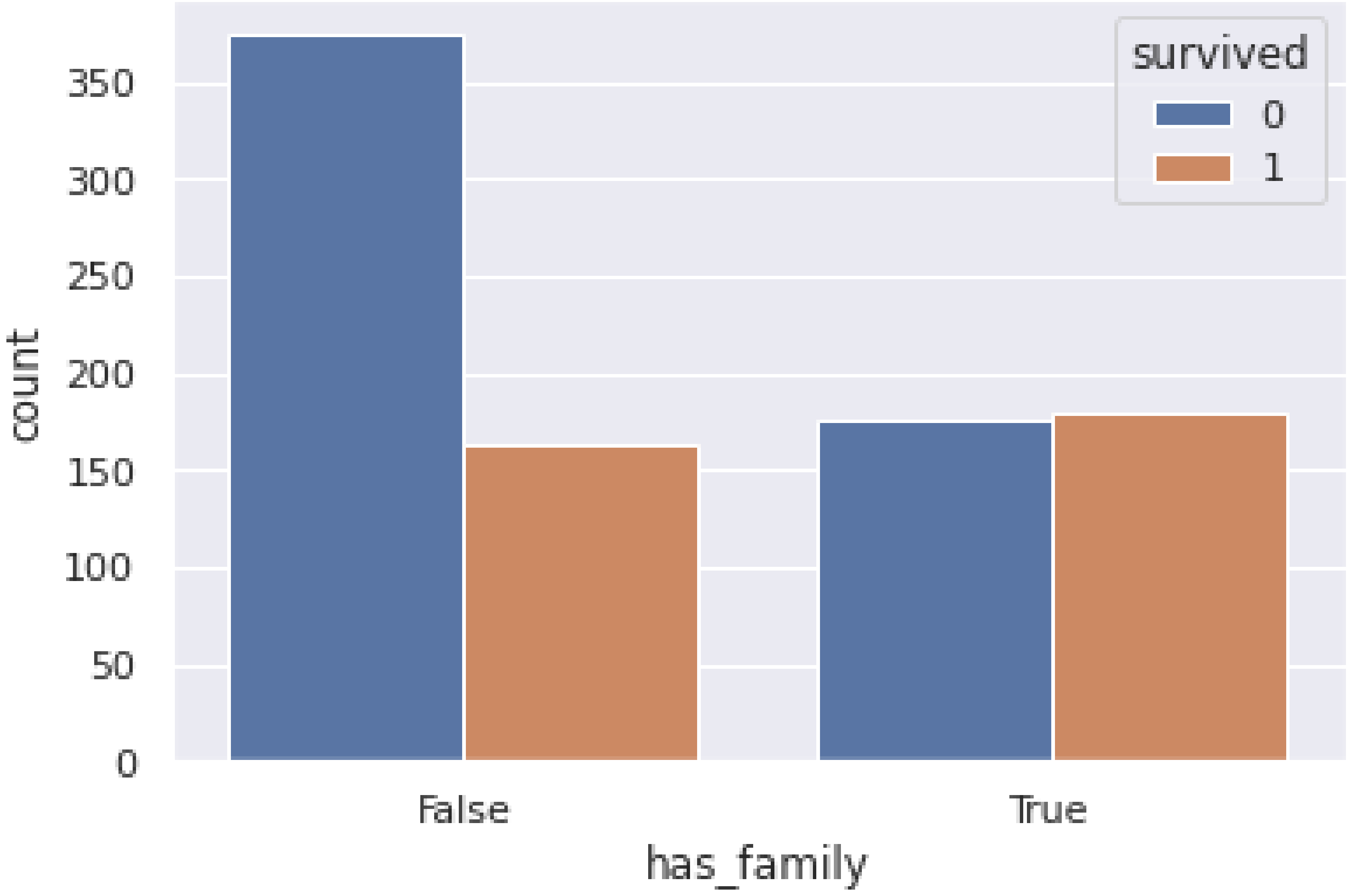
4.4 People with families had a better chance of survival? (CONT.)

```
def hasFamily(family):
    if family >= 1:
        return True
    else:
        return False

has_parent_children =train_data.parent_children.apply(hasFamily)
has_sibling_spouse =train_data.sibling_spouse.apply(hasFamily)
train_data['has_family'] = has_sibling_spouse | has_parent_children
```

```
data =sns.countplot("has_family", data=train_data, hue="survived")
```

4.4 People with families had a better chance of survival?



4.5 Drop columns that was made for analysing

```
#drop culomns that was made for analysing
train_data.drop(labels = ["has_family", "age_group"], axis=1, inplace = True)
```

STEP 5:
Data Modeling
& Predicition

5.1 Split the train data into input features and target

```
# since the test_data doesnt contain the 'survived' column we can't test the accurcy of the
# model so we will divide the train data to two sets to train and test the model

input_features = train_data.drop(columns = ['survived'],axis=1)
target = train_data['survived']

# Now we will split the input features into two sets and the target as well
# train the model with input_features_train and target_train
# The model predicts the output using input_features_test
# test the accuracy of the model using predicit values and target_test

input_features_train, input_features_test, target_train ,target_test = train_test_split(inpu
t_features,target,test_size=0.3)
```

5.2 Build the model

```
# input features
print(input_features.columns)
```

```
Index(['passenger_id', 'passenger_class', 'sex', 'age', 'sibling_spouse',
       'parent_children', 'fare', 'embarked'],
      dtype='object')
```

```
# build the model using Logistic Regression
model = LogisticRegression(solver='liblinear')
model.fit(input_features_train, target_train)
```

```
LogisticRegression(solver='liblinear')
```

5.3 Predict the target and test accuracy

```python
# predict the output of the testing dataset
predict = model.predict(input_features_test)
print(predict)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1
 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1
 1 0 1 0 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1
 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 0 1
 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0
 1 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0
 0 1 1 0 0 1 0 0 0]
```

```python
# check how accurate was its prediction
accuracy = accuracy_score(target_test, predict)
print('Accuracy : ',accuracy)
```

```
Accuracy :  0.8246268656716418
```

STEP 6: Performance metrics

6.1 Confusion Matrix

```
#
confusion_matrix(target_test, predict)
```

```
array([[147,  15],
       [ 32,  74]])
```

6.2 Testing survived and predict

```
#
print(classification_report(target_test, predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       162
           1       0.83      0.70      0.76       106

    accuracy                           0.82       268
   macro avg       0.83      0.80      0.81       268
weighted avg       0.83      0.82      0.82       268
```

STEP 7:
Process for Submission File

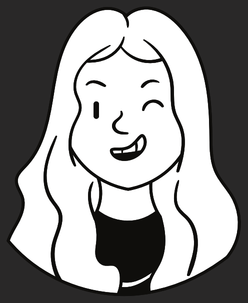
```python
# predect the values for the test_data for the compatition
prediction = model.predict(test_data)
test_data["survived"] = prediction
test_data.drop(labels = ["passenger_class", "sex","age","sibling_spouse", "parent_children",
"fare","embarked"], axis=1, inplace = True)
test_data.head()
```

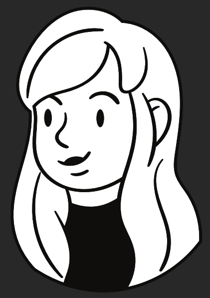
	passenger_id	survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1

OUR GROUP


Najla Alshehri


Reema Faisal


Fay Almanea


Fatimah Almutab


Lma Alhazmi

THANK YOU