Titanic

Machine Learning from Disaster

MTD09 team: Suresh Kumar Shanmuga Sundaram

Abstract:

The RMS Titanic was a British passenger liner that sank in the North Atlantic Ocean in the early morning hours of 15 April 1912, after it collided with an iceberg during its maiden voyage from Southampton to New York City. The broader goal is to provide other aspiring data scientists when analyzing new data. with cleanly coded view of data analysis. The plan is to explain topics so that people can understand my thought process and the general flow that I use when analyzing new data.

ANALYSIS OF PROJECT:

We will be analyzing Titanic data which contains demographics and passenger information that whether they survived or died

The analysis of Titanic data is shown below

Importing the Libraries

To install the necessary libraries for data analysis, you can use the `pip` command. Open your terminal or command prompt and run the following commands:

```
Step 1: Import necessary libraries

pip install pandas seaborn matplotlib

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Getting the Data

```
# Step 2: Load the Titanic dataset
titanic = sns.load_dataset('titanic')

# Display the first few rows of the dataset
print(titanic.head())
```

Data Exploration/Analysis

```
print(titanic.info())
```

✓ 0.0s

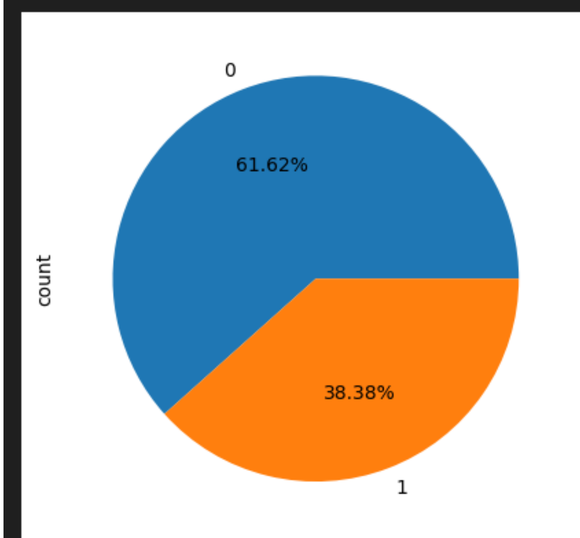
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

This Dataset has 891 examples and 14 features + the target variable (survived). 2 of the features are bool, 2 are Category, 2 of the features are floats, 4 are integers and 5 are objects. Below I have listed the features with a short description:

```
survival: Survival
PassengerId: Unique Id of a passenger.
pclass: Ticket class
sex: Sex
Age: Age in years
sibsp: # of siblings / spouses aboard the Titanic
parch: # of parents / children aboard the Titanic
ticket: Ticket number
fare: Passenger fare
cabin: Cabin number
embarked: Port of Embarkation
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic['survived'].value_counts().plot.pie(autopct='%1.2f%%')
✓ 0.0s
<Axes: ylabel='count'>
```



Above we can see that **38% out of the training-set survived the Titanic**. We can also see that the passenger ages range from 0.4 to 80. On top of that we can already detect some features, that contain missing values, like the ‘Age’ feature.

```
titanic.head(8)
```

[78] ✓ 0.0s

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	no	True
5	0	3	male	28.0	0	0	8.4583	Q	Third	man	True	no	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	no	True
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	no	False

From the table above, we can note a few things. First of all, that we **need to convert a lot of features into numeric** ones later on, so that the machine learning algorithms can process them. Furthermore, we can see that the **features have widely different ranges**, that we will need to convert into roughly the same scale. We can also spot some more features, that contain missing values (NaN = not a number), that we need to deal with.

Survived (int to Category)

Pclass (int to Category)

Sex(object to Category)

Age(float to int)

Embarked(object to Category)

```
print(titanic.isnull().sum())
✓ 0.0s

survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

The Embarked feature has only 2 missing values, which can easily be filled. It will be much more tricky, to deal with the ‘Age’ feature, which has 177 missing values. The ‘Cabin’ feature needs further investigation, but it looks like that we might want to drop it from the dataset, since 77 % of its data are missing.

Imputing missing values for ‘Age’

Imputing is a technique for replacing the missing data with same substitute value, strategy that we are using here is mean value.

Using mode by finding the most appeared value in embarked Column

And dropping the deck and ‘embark_ town’ as they have many missing values

```python
# Fill missing values for 'age' with the median value
titanic['age'].fillna(titanic['age'].median(), inplace=True)

# Fill missing values for 'embarked' with the mode
titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)

# Drop columns 'deck' and 'embark_town' as they have many missing values
titanic.drop(columns=['deck', 'embark_town'], inplace=True)
```

At this point, we have taken care of all the missing data that we had.
Now we are changing the data type of the following columns using the as type method since they were inappropriate

After changing all necessary steps,

Now there are no missing values.
We have changed the data type of the columns that had inappropriate data types.
Our memory usage has also reduced

```python
titanic.info()
```

✓ 0.0s

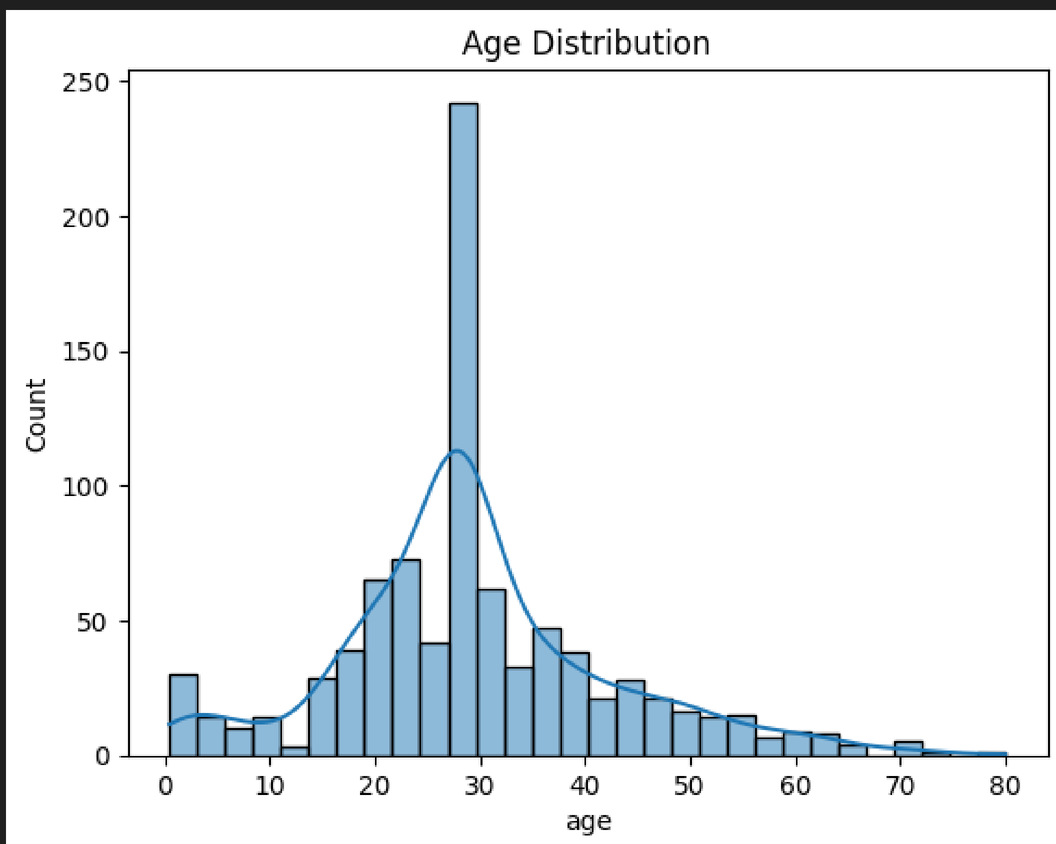
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   survived    891 non-null    int64
 1   pclass      891 non-null    int64
 2   sex         891 non-null    object
 3   age         891 non-null    float64
 4   sibsp       891 non-null    int64
 5   parch       891 non-null    int64
 6   fare        891 non-null    float64
 7   embarked    891 non-null    object
 8   class       891 non-null    category
 9   who         891 non-null    object
 10  adult_male  891 non-null    bool
 11  alive       891 non-null    object
 12  alone       891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(4)
memory usage: 72.5+ KB
```

Above you can see the 12 features + the target variable (survived). **What features could contribute to a high survival rate ?**

1. Age Distribution of Passengers :

```
# 4.4. Age Distribution
# Age distribution
sea.histplot(titanic['age'], bins=30, kde=True)
plt.title('Age Distribution')
plt.show()
```

✓ 0.1s

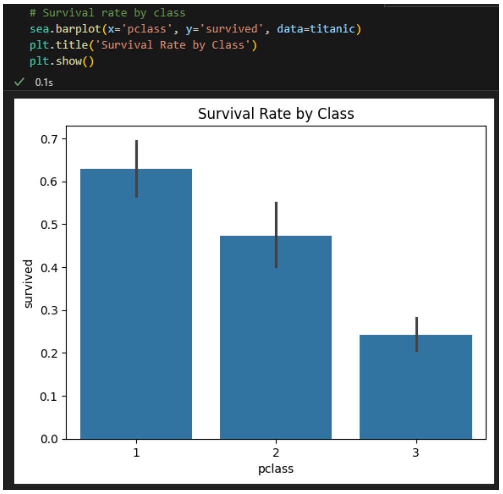


Observation :

By looking at the displot we can say that most of the

Passengers were in between the age range of 20 40.

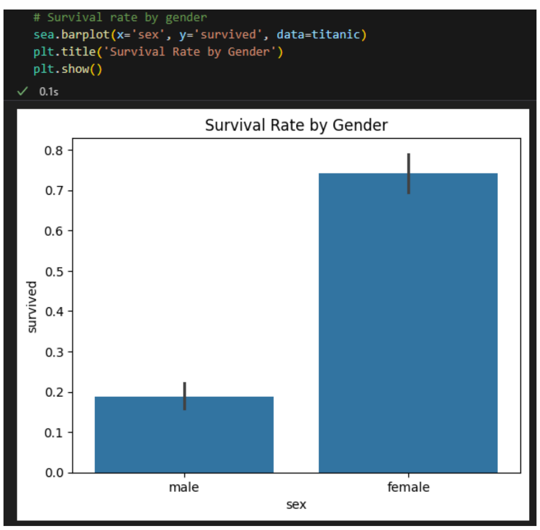
2. Survival rate by Class



Observation:

Here we see clearly, that Pclass is contributing to a person's chance of survival, especially if this person is in class 1.

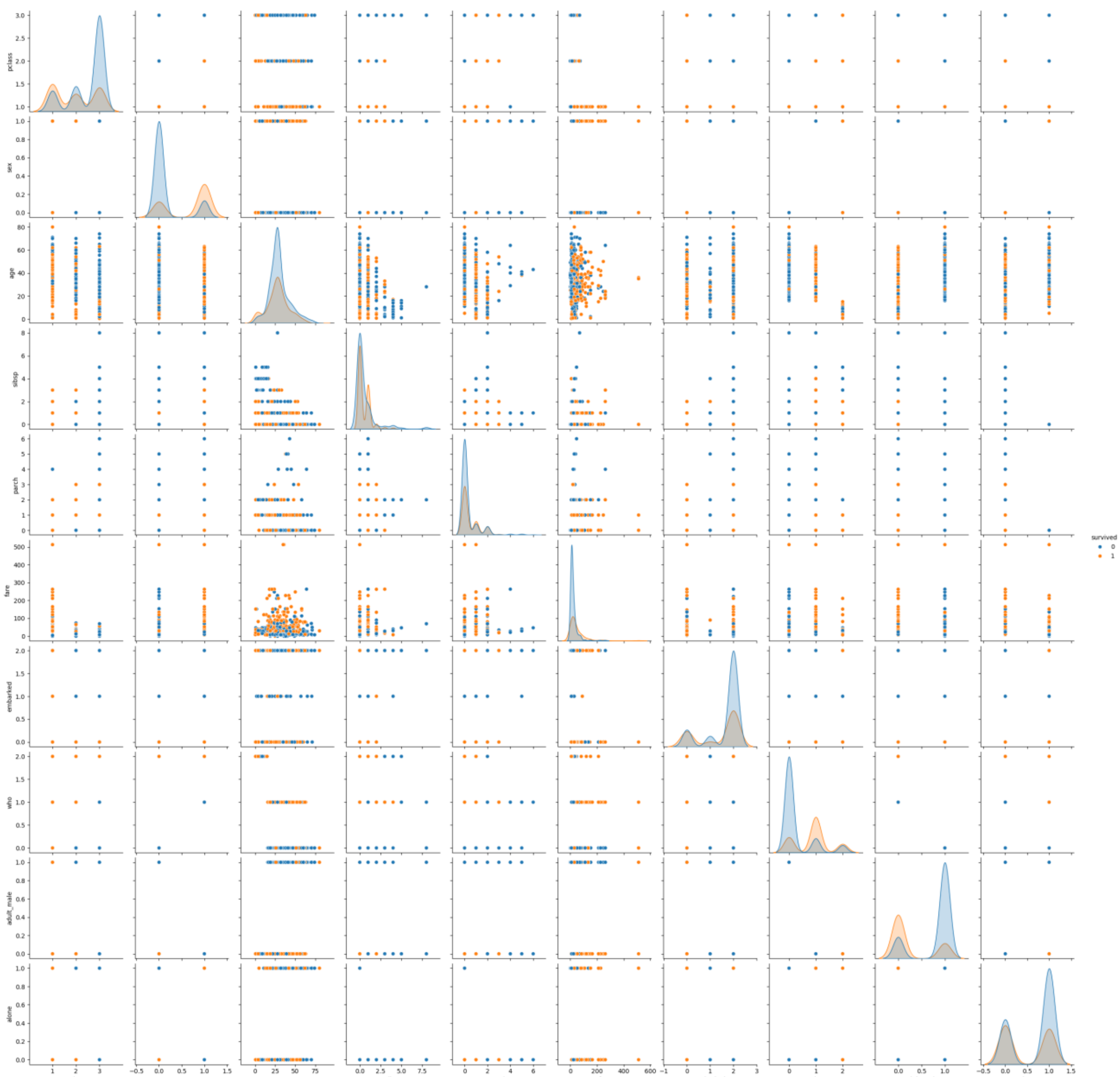
3. Survival rate by Gender



It shows the survival count of male and female ratio. Out of hundred percentages 65 are female and 35 are male.

4. Pairplot:

```
# Step 6: Pair Plot
# Pair plot
sea.pairplot(titanic_encoded, hue='survived')
plt.show()
```



5. Conclusion:

- Positively correlated with `Sex` (0.54), indicating that females were more likely to survive.
- Positively correlated with `Fare` (0.26), suggesting that higher fares (often linked with higher class) had a higher survival rate.
- Negatively correlated with `Pclass` (-0.34), indicating that passengers in lower classes had lower survival rates.
- Negatively correlated with `Alone` (-0.2), indicating that passengers who were alone had a slightly lower survival rate.