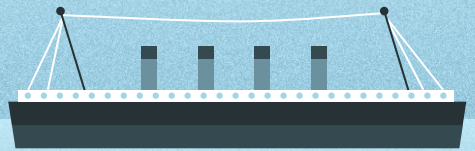
— Who Will Survive? —

TITANIC

Ammar Ali and Guhfran Shuhood



EDA ⟶ Handling Missing/Null Entries

My first step of EDA was dealing with any potential missing values. I started by printing out the sum of all null entries:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Embarked: any entry that had a missing value for embarked was dropped, since only 2 were missing.

Age: Mean imputation, was representative of data because mean ≈ median.

Fare: 1 missing value (Mean imputation)

Cabin: Imputed “N/A”, most of this column was missing, and there’s no good way to predict it.

EDA → Handling Outliers

Using the describe feature of dataframe objects allowed me to get a good feel of what my data looked like. This allowed me to get a quick look at some of the outliers.

Age: Kept age outliers because they were still realistic (80 year is realistic, unlike 250 years)

Fare: Kept all of the fare data because it can provide insightful information and reveal trends about survival chances (ex. Are more expensive tickets going to increase your survival odds?).

```
Describing data
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

             Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

EDA ⟶ Handling Non-Numerical Data

1.

Embarked

Each port was represented numerically (0-2) in the order in which they were departed from, possible because it was ordinal.

```
trainingdf['Embarked_numeric'] = trainingdf['Embarked'].map({'S': 0, 'C':1, 'Q':2})
```

2.

Sex

Simply representing the data using binary digits 0 and 1

```
trainingdf['Sex_binary'] = trainingdf['Sex'].map({'male': 0, 'female': 1})
```

Data Left Out from Correlation Analysis

1.

Name

- Cannot convert to numerical form
- Unique per passenger
- Cannot ‘intrinsically’ increase survival odds

2.

Ticket

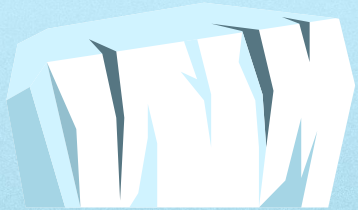
- Cannot convert to numerical form
- No useful information

3.

Cabin

- Too much missing data/undescriptive
- Unique per passenger

4.



PassengerID

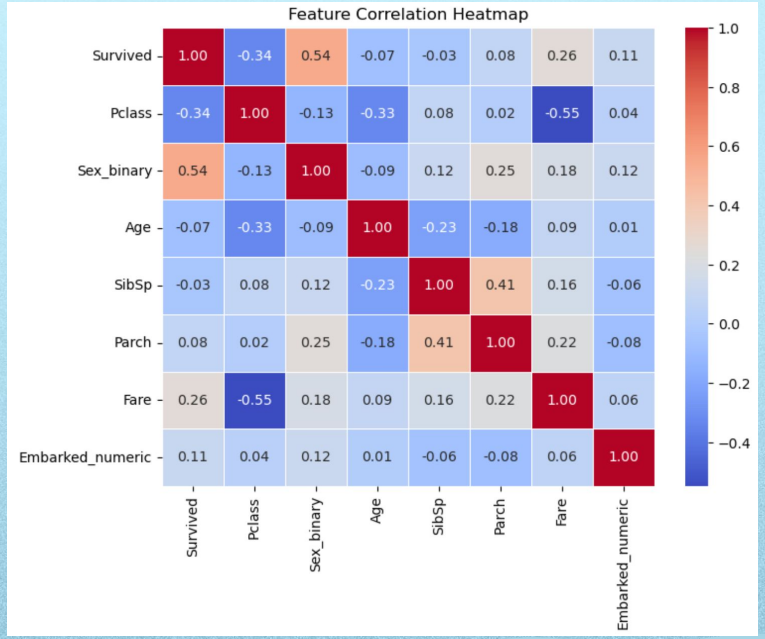
- Cannot ‘intrinsically’ increase survival odds
- Unique per passenger

Pearson Correlation Analysis

Sex and Survived (0.54): somewhat strong positive correlation to being female and surviving, begging the following questions: Were women given priority when trying to get passengers to safety, or did they have some superior survival instincts?

Survived and pclass (-0.34): Better class tickets were associated with better chances of survival, suggesting that first class/more expensive tickets were safer.

Age and pclass (-0.33): This revealed a socioeconomic trend, implying that there is a moderate negative correlation with age and the class of their ticket, suggesting older people were more likely to buy first class tickets than younger people. Factors such as gender however did not play a role here.

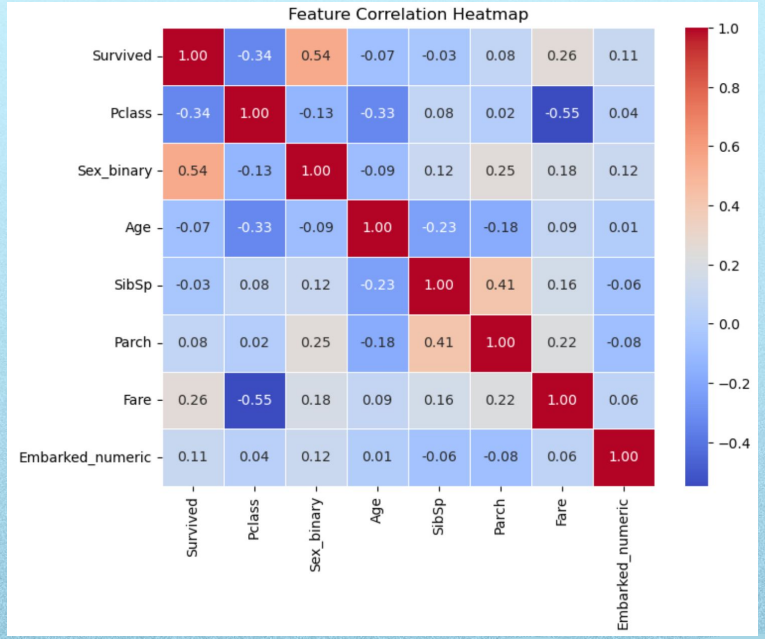


Pearson Correlation Analysis cont.

Sex and Parch (0.25): implies that women are more likely to have parents/children on board. Can this be related back to the higher survival odds for women, implying traveling with family is safer?

Age and Parch (-0.18): older passengers had fewer parents/children on board. This makes sense because as you get older, you are more capable to go on trips with fewer people. Children on the other hand are very likely to be accompanied by a parent.

Sex and fare (0.18): women are more likely to pay slightly more for their tickets than men. It was found earlier that they have higher survival odds, and that a better class ticket also has higher survival odds. This may explain it, suggesting that women survived more due to purchasing the better tickets.



Modeling

- Split train.csv into 80% training and 20% testing data
- Randomly chose 80% and 20% fields respectively
- Only concerned with tangible numerical fields that can give us values for modeling

```
# Only use the numerical values
sample_df = trainingdf[['Survived', 'Pclass', 'Sex_binary', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_numeric']]

# Randomly choose 80% for training and other 20% for testing

# Sample 712 random rows (80%) with popular seed value 42
train_df = sample_df.sample(n=712, random_state=42)

# The remaining rows go into the second subset
test_df = sample_df.drop(train_df.index)
```

Model 1 - Logistic Regression (All Numeric Features)

- 1000 Iterations
- Only used numerical features (converted gender and embarked to numeric values)

```
x_train = train_df[['Pclass', 'Sex_binary', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_numeric']]
y_train = train_df[['Survived']]

x_test = test_df[['Pclass', 'Sex_binary', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_numeric']]
y_test = test_df[['Survived']]

logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(x_train, y_train)
logistic_predictions = logistic_model.predict(x_test)

logistic_f1_score = f1_score(y_test, logistic_predictions)
print(f"Logistic Regression F1_score: {logistic_f1_score}")
```

```
Logistic Regression F1_score: 0.6446280991735538
```

Model 2 - K Nearest Neighbour

Best Performing Hyperparameters with F1 Scores:

k = 3 (0.732) and k = 8 (0.703)

```python
# Feature Scaling is required
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

```python
# Experiment with Hyperparameter tuning for k
for i in range(1,10):
    # Initialize KNN (start with k=i as a default)
    knn = KNeighborsClassifier(n_neighbors=i)

    # Train
    knn.fit(x_train_scaled, y_train)

    # Predict
    knn_predictions = knn.predict(x_test_scaled)

    # Evaluate
    knn_f1 = f1_score(y_test, knn_predictions)
    print(f"KNN F1 Score k={i}: {knn_f1}")
```

```
KNN F1 Score k=1: 0.6666666666666666
KNN F1 Score k=2: 0.6857142857142857
KNN F1 Score k=3: 0.7226890756302522
KNN F1 Score k=4: 0.6923076923076923
KNN F1 Score k=5: 0.6956521739130433
KNN F1 Score k=6: 0.6972477064220183
KNN F1 Score k=7: 0.6837606837606838
KNN F1 Score k=8: 0.7027027027027027
KNN F1 Score k=9: 0.6842105263157895
```

Model 3 - Logistic Regression (Selected Features)

Best Candidate uses features: **Pclass, Sex_binary, Parch,** and **SibSp**

Best F1_score is **0.6842**

```
# Without Pclass: 0.6495726495726496
# Without Age: 0.6610169491525425
# Without Sex_binary: 0.4999999999999999 !!!!
# Without SibSp: 0.65
# Without Parch: 0.65
# Without Fare: 0.65
# Without Embarked_numeric: 0.639344262295082
# Without Age & Embarked_numeric: 0.6724137931034484
# Without Age, Embarked_numeric, Fare: 0.6842105263157895
# Without Pclass, Age, Fare, Embarked_numeric: 0.6724137931034484
```

Answer

- Best Classifier is KNN with hyperparameter k set to 3
- Used train.csv as the training set and test.csv as the testing set
- Fixed test.csv
  - Used mean to impute the only missing fare record (only 1 row)
  - Used same strategies from train.csv to fix Age and Cabin fields
- Generated csv dump "cse351_project_prediction.csv" of 2 columns:
  - Passenger ID
  - Survived (either 0 or 1)

—Contributions—

1. EDA → Ammar Ali

2. Modeling → Guhfran Shuhood

Both worked on slides & report