CS675 FINAL PROJECT

Ibrahim Mohammed Hamed - Lolyna de la Fuente - Chukwuebuka Ozoh

AGENDA

- EXPLORATORY DATA ANALYSIS (EDA)
- DATA PREPROCESSING
- FEATURE ENGINEERING
- CONDITIONAL VISUALIZATIONS
- MODELS FOR ANALYSIS
- CONCLUSION

ABOUT THE DATABASE

SHAPE

891 ROWS X 12 COLUMNS

7 NUMERICAL

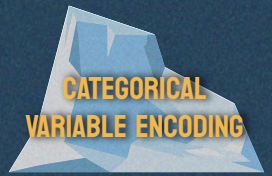
- PASSENGERID
- SURVIVED
- PCLASS
- AGE
- SIBSP
- PARCH
- FARE

5 CATEGORICAL

- NAME
- SEX
- TICKET
- CABIN
- EMBARKED

DATA PREPROCESSING

We had 866 Null Values in columns Age, Cabin and Embarked.
For Age we replaced the NaNs with the mean.
For Cabin we removed the column as it had 687/891 rows with NaNs.
For Embarked we replaced the NaNs with mode.



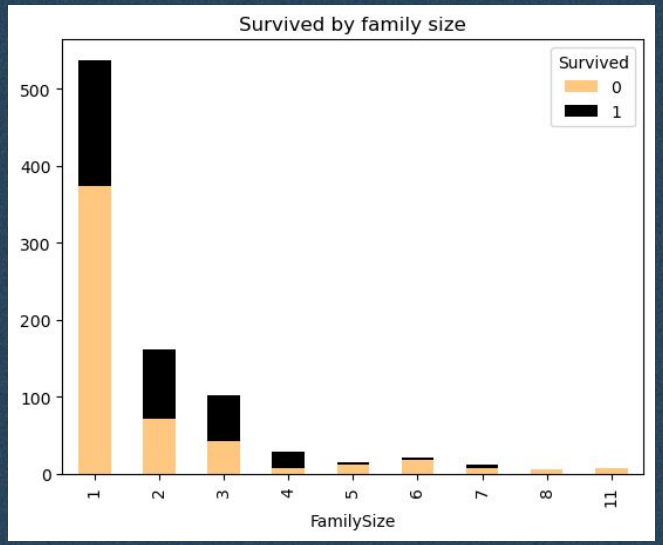
We had 4 categorical categories; Sex, Embarked, Name and Ticket.
We did one-Hot Encoding for Sex and label Encoder for Embarked columns.

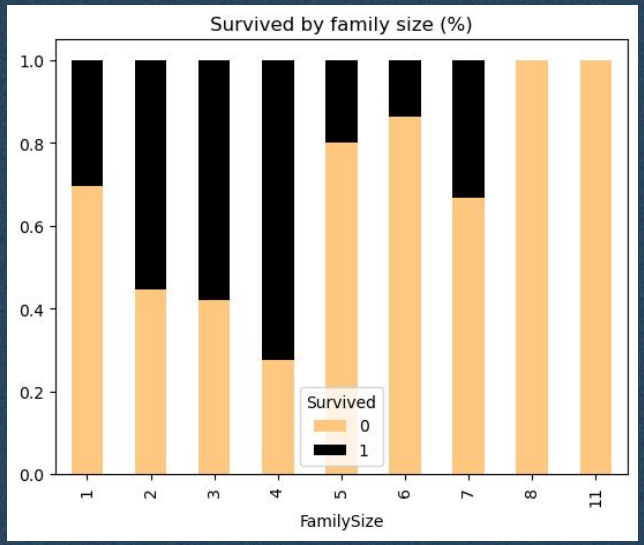
We used StandardScaler for Age and Fare columns as they were the only ones with values above 3.

FEATURE ENGINEERING

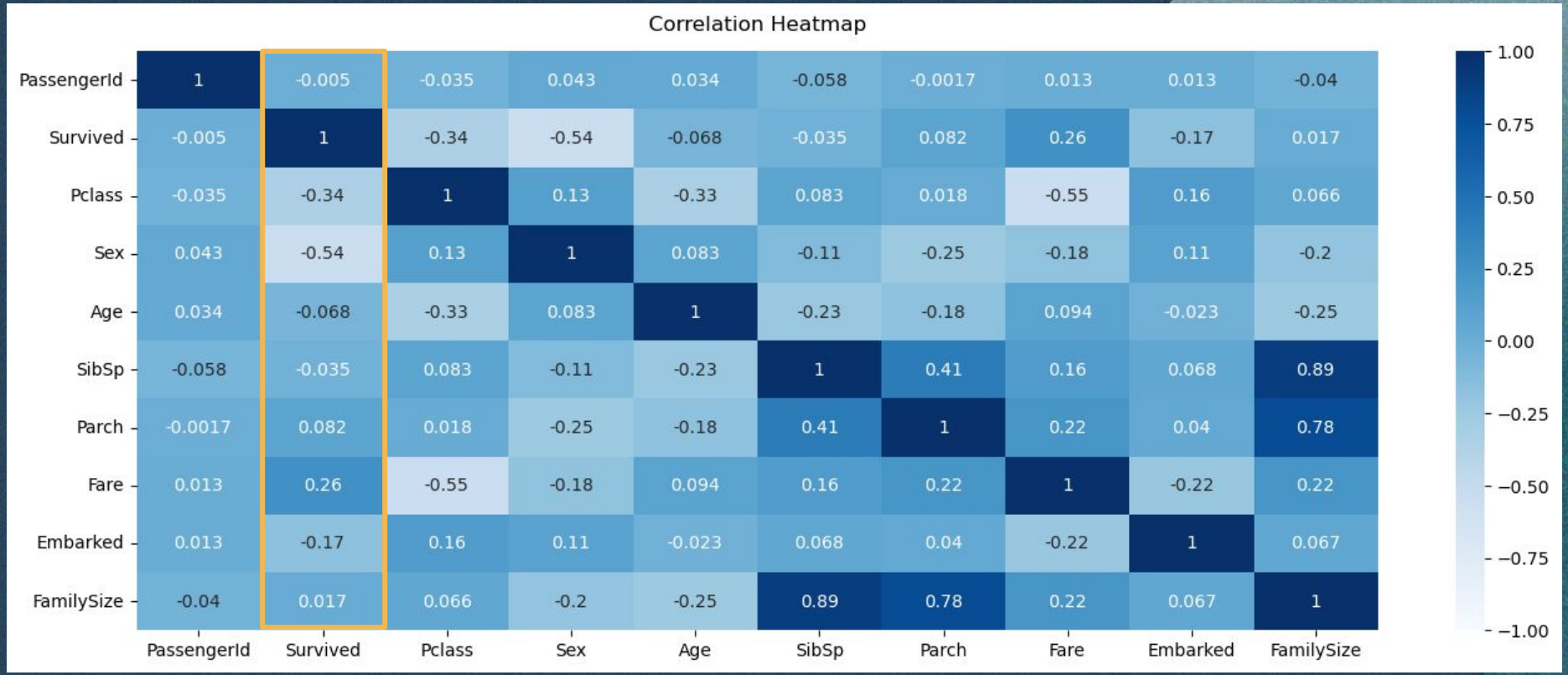
CREATED A COLUMN FOR FAMILY SIZE

SUM OF SIBSP AND PARCH COLUMNS

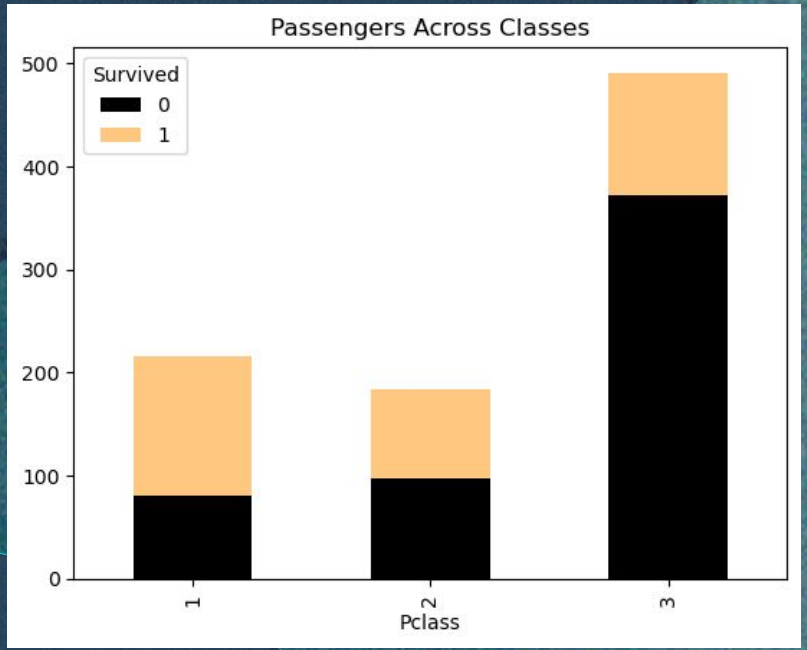


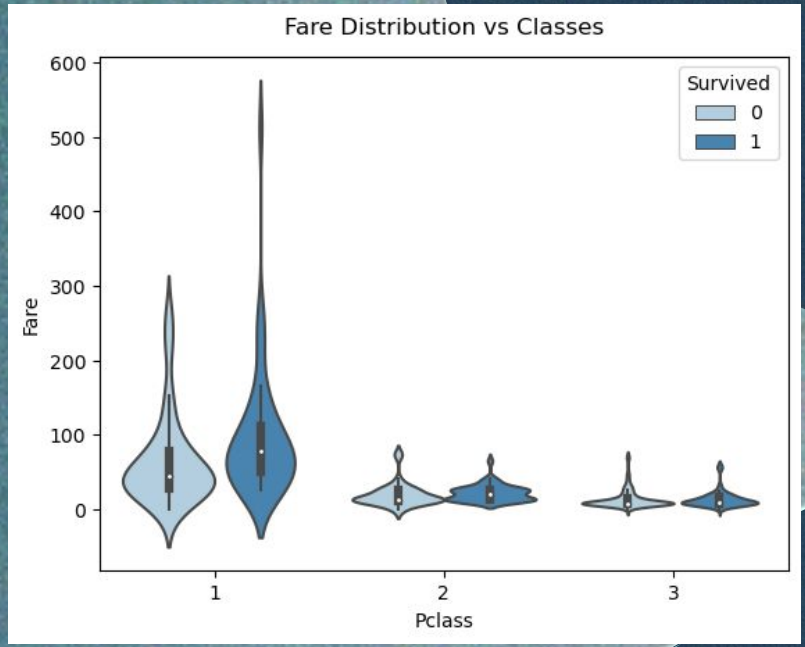


CONDITIONAL VISUALIZATIONS

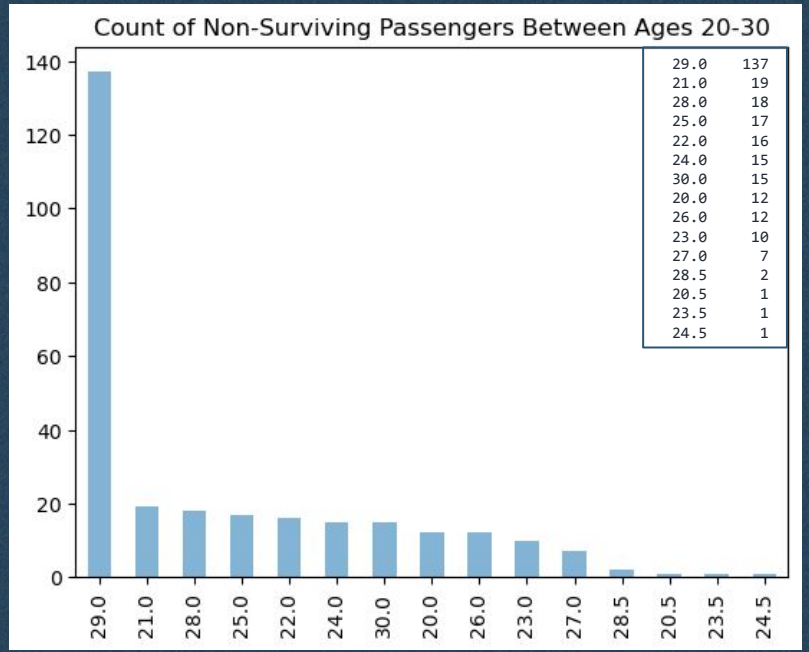


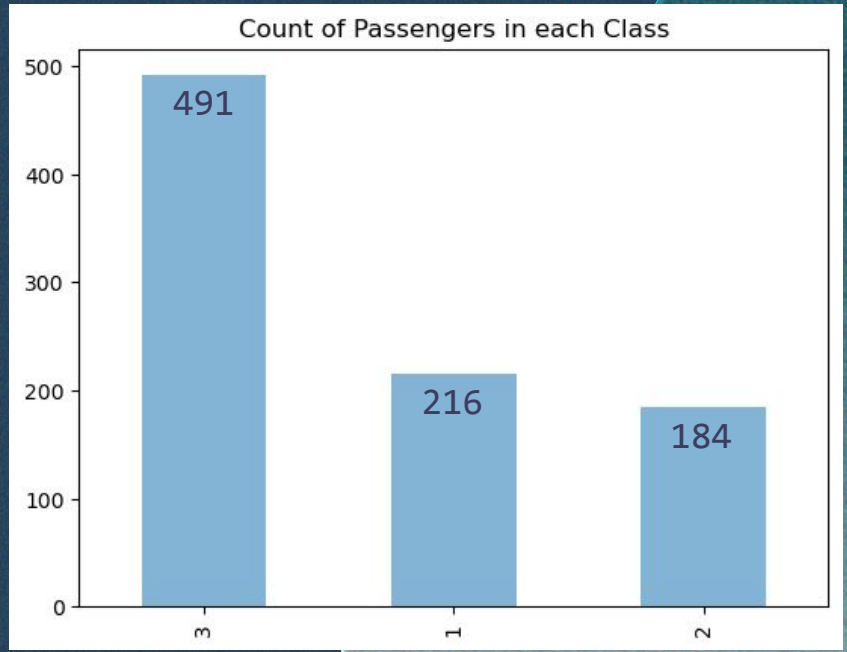
CONDITIONAL VISUALIZATIONS





CONDITIONAL VISUALIZATIONS





MODELS FOR ANALYSIS

TRAIN AND TEST SPLIT

```
# Split the dataset into training and testing sets.
X = sdf.iloc[:,[2,4,5,6,7,9,10,11]]
y = sdf.iloc[:,1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

X

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize
0	3	1	-0.581659	1	0	-0.502445	2	2
1	1	0	0.649327	1	0	0.786845	0	2
2	3	0	-0.273913	0	0	-0.488854	2	1
3	1	0	0.418517	1	0	0.420730	2	2
4	3	1	0.418517	0	0	-0.486337	2	1

Y

WE USED THE SURVIVE COLUMN:

- 0 = DID NOT SURVIVED
- I = SURVIVED

MODELS FOR ANALYSIS

RANDOM FOREST

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```
✓ 0.0s

```
rf_classifier.fit(X_train, y_train)
```
✓ 0.1s

```
RandomForestClassifier(random_state=42)
```

```
y_pred_rf = rf_classifier.predict(X_test)
```
✓ 0.0s

```
accuracy = accuracy_score(y_test, y_pred_rf)
print("Accuracy:", accuracy)
```
✓ 0.0s

```
Accuracy: 0.7873134328358209
```

DECISION TREE

```
dt_classifier = DecisionTreeClassifier(random_state=42)
```
✓ 0.0s

```
dt_classifier.fit(X_train, y_train)
```
✓ 0.0s

```
DecisionTreeClassifier(random_state=42)
```

```
y_pred_dt = dt_classifier.predict(X_test)
```
✓ 0.0s

```
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", accuracy_dt)
```
✓ 0.0s

```
Decision Tree Accuracy: 0.7350746268656716
```

RANDOM FOREST

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.80	0.85	0.82	157
1	0.76	0.70	0.73	111
accuracy			0.79	268
macro avg	0.78	0.77	0.78	268
weighted avg	0.79	0.79	0.79	268

The Random Forest model achieved an overall accuracy of 79% on the test dataset. It performed well in classifying the negative class (0) with a precision of 80% and a recall of 85%.

However, its performance on the positive class (1) was slightly lower, with a precision of 76% and a recall of 70%.

The model's F1-score, which balances precision and recall, was 0.82 for the negative class and 0.73 for the positive class.

In summary, the Random Forest model demonstrated a balanced performance in classifying both classes, with slightly higher accuracy in the negative class.

DECISION TREE

Decision Tree Classification Report:

	precision	recall	f1-score	support
0	0.77	0.79	0.78	157
1	0.69	0.66	0.67	111
accuracy			0.74	268
macro avg	0.73	0.72	0.73	268
weighted avg	0.73	0.74	0.73	268

On the other hand, The Decision Tree model achieved an overall accuracy of 74% on the test dataset.

It demonstrated a slightly lower performance compared to the Random Forest model, with a precision of 77% for the negative class (0) and a recall of 79%.

Similarly, for the positive class (1), the precision was 69% and the recall was 66%.

The F1-score, which balances precision and recall, was 0.78 for the negative class and 0.67 for the positive class.

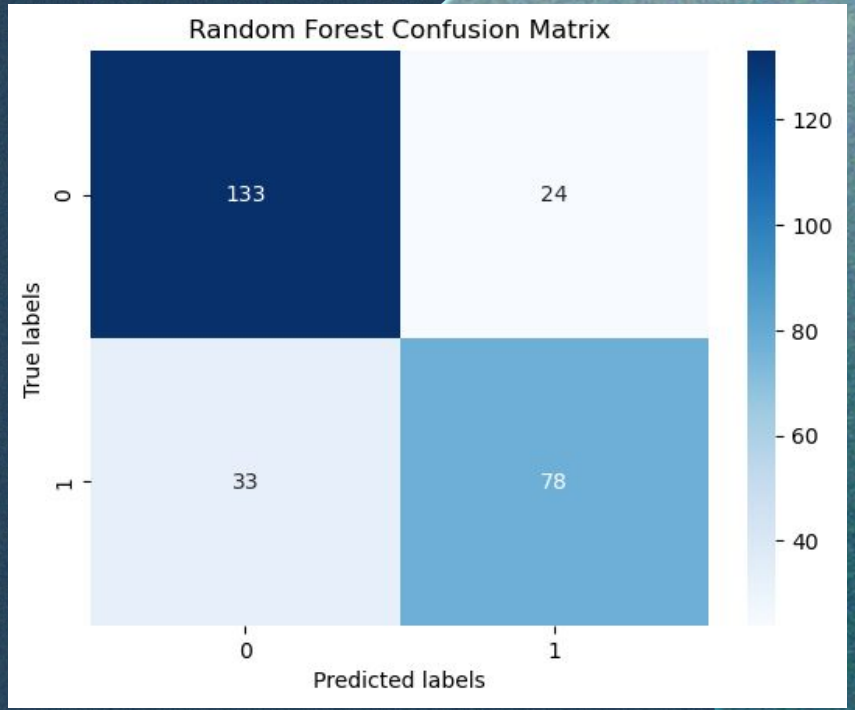
Overall, the Decision Tree model showed a reasonable performance in classifying both classes, although not as balanced as the Random Forest model.

RANDOM FOREST CONFUSION MATRIX

The confusion matrix provides a detailed breakdown of the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.

In the case of our Random Forest model, we can see that out of 268 samples in the test set, it correctly predicted 133 instances of survival (true positives) and 110 instances of non-survival (true negatives).

However, it misclassified 24 instances as survived when they did not (false positives), and 21 instances as non-survived when they did (false negatives).



MODELS FOR ANALYSIS

LOGISTIC REGRESSION

```
#Model Training
logreg = LogisticRegression(random_state=42)
logreg.fit(X_train, y_train)
```

✓ 0.0s

```
LogisticRegression(random_state=42)
```

```
# Model Evaluation
y_pred_lr = logreg.predict(X_test)
```

✓ 0.0s

```
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
```

✓ 0.0s

```
Accuracy: 0.8134328358208955
```

K-NEAREST NEIGHBORS

```
# Initialize and train the KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

✓ 0.0s

```
KNeighborsClassifier()
```

```
# KNN Predictions
knn_predictions = knn_model.predict(X_test)
```

✓ 0.0s

```
# Calculate the accuracy for KNN
knn_accuracy = accuracy_score(y_test, knn_predictions)
print(f'KNN Accuracy: {knn_accuracy:.2f}')
```

✓ 0.0s

```
KNN Accuracy: 0.75
```

LOGISTIC REGRESSION

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.82	0.87	0.85	157
1	0.80	0.73	0.76	111
accuracy			0.81	268
macro avg	0.81	0.80	0.80	268
weighted avg	0.81	0.81	0.81	268

The Logistic Regression model achieved an overall accuracy of 81% on the test dataset. It performed well in classifying the negative class (0) with a precision of 82% and a recall of 87%.

However, its performance on the positive class (1) was slightly lower, with a precision of 80% and a recall of 73%.

The model's F1-score, which balances precision and recall, was 0.85 for the negative class and 0.76 for the positive class.

In summary, the Logistic Regression model demonstrated a balanced performance in classifying both classes, with slightly higher accuracy in the negative class.

K-NEAREST NEIGHBORS

KNN Classification Report:

	precision	recall	f1-score	support
0	0.75	0.87	0.80	157
1	0.76	0.59	0.67	111
accuracy			0.75	268
macro avg	0.76	0.73	0.74	268
weighted avg	0.75	0.75	0.75	268

The K-Nearest Neighbors model achieved an overall accuracy of 75% on the test dataset.

It demonstrated a slightly lower performance compared to the Logistic Regression model, with a precision of 75% for the negative class (0) and a recall of 87%.

The positive class (1) had a precision of 76% and a recall of 59%.

The model's F1-score, which balances precision and recall, was 0.80 for the negative class and 0.67 for the positive class.

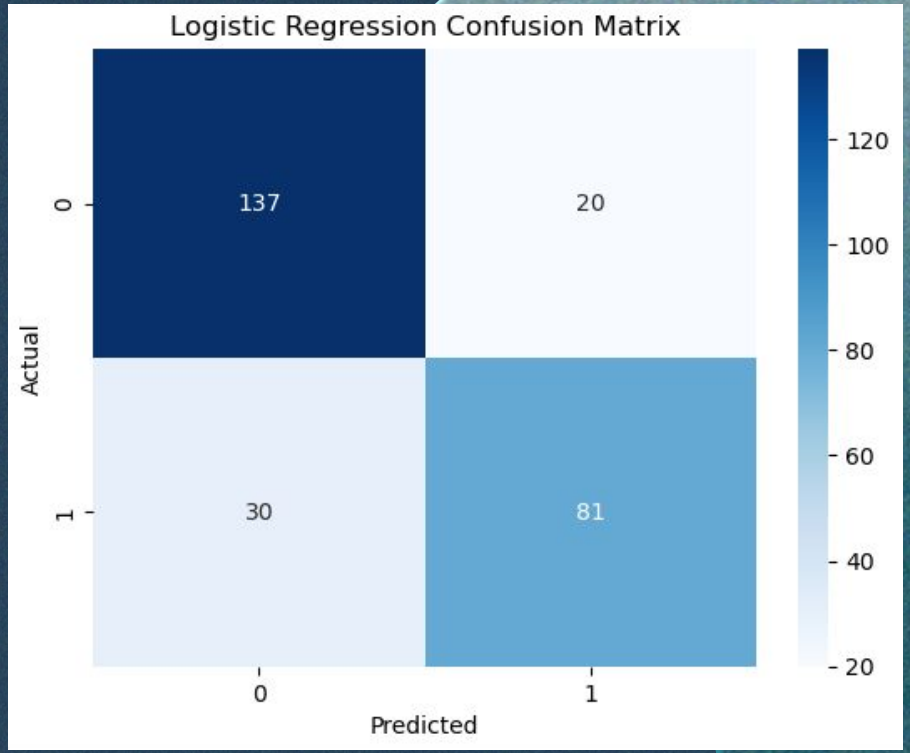
Overall, the K-Nearest Neighbors model showed a reasonable performance in classifying both classes.

LOGISTIC REGRESSION CONFUSION MATRIX

The confusion matrix provides a detailed breakdown of the model's performance by showing the number of true positives, true negatives, false positives, and false negatives.

For the Logistic Regression Model, we see that out of 268 samples in the test set, it correctly predicted 137 instances of survival (true positives) and 110 instances of non-survival (true negatives).

However, it misclassified 20 instances as survived when they did not (false positives), and 31 instances as non-survived when they did (false negatives).



CONCLUSION

The analysis of the Titanic Dataset provided valuable insights into factors influencing passenger survival. Through data exploration, preprocessing, and modeling techniques such as Random Forest, Decision Trees, KNN, and Logistic Regression, we gained a deeper understanding of the dataset's structure and predictive capabilities.

Key Findings include the importance of certain features like gender, age, and passenger class in predicting survival probabilities. Furthermore, the evaluation of model performance through metrics like precision, recall, and accuracy allowed us to assess the effectiveness of different algorithms in predicting survival outcomes.

Overall, this project demonstrated the power of data analysis and Machine Learning in uncovering patterns and making predictions from historical data.

THANKS!

Slide Chef