

CS 4650.01 – Big Data and Cloud Computing

Capstone – Final - Report

1) Who is on your team?

Our team consists of 3 members.

- Pau Khai
- Shreyas Chaudhary
- Vikram Ramesh

2) Which dataset/competition did you choose?

- We chose the following competition: **Titanic –Machine Learning from Disaster**
- Link: <https://www.kaggle.com/competitions/titanic>

The Titanic dataset includes the following key features:

- Passenger Class: A proxy for socio-economic status (1st = Upper class, 2nd = Middle class, 3rd = Lower class).
- Name, Sex, Age: Basic demographics.
- SibSp: Number of siblings/spouses aboard.
- Parch: Number of parents/children aboard.
- Ticket Number, Fare: Details about the journey.
- Cabin: Cabin number which might infer location on the ship.
- Embarked: Port of embarkation.

3) What approach did you take? What steps did you take to clean the data, improve the results, etc.

- **Approach:**
- **For cleaning the data, we did the following:**

To clean the data, initially, we addressed missing values by filling the 'Embarked' feature with the most common embarkation point and replacing missing 'Fare' values in the test dataset with the median 'Fare' from the training dataset. Recognizing the large amount of missing 'Age' data, we employed a predictive approach based on extracted titles from passenger names, which were standardized and used to estimate ages by the mode within title groups. Furthermore, we streamlined the dataset by removing redundant features. The 'Fare' feature was dropped after introducing 'FareBand', and similar reductions were made by removing the 'Name', 'Cabin',

'Ticket', and continuous 'Age' features, replacing them with more analytical-friendly categorical data. These steps have prepared our dataset for more effective model training and analysis.

Steps Taken:

1. Data Loading and Exploration:

- Loaded the Titanic dataset
- Explored the initial data structure to check types and content:

`train.head()` to preview data

`train.describe(include="all")` to get a summary of all columns, including non-numeric.

2. Initial Data Analysis:

- Identified and analyzed the dataset to understand the types and completeness of features:

Total of 891 passengers in the training set.

Numerical features like Age (19.8% missing) and Fare.

Categorical features like Sex, Embarked (0.22% missing), and Pclass.

Alphanumeric features like Cabin (77.1% missing) and Ticket.

3. Data Cleaning:

- Addressed missing values as explained above

4. Feature Engineering:

Title Extraction: From 'Name', extracting titles and mapping them to numerical codes, providing social status insight.

Family Size: Combined 'SibSp' + 'Parch' + 1 to capture family dynamics onboard.

Age Groups: Binned ages into categories such as Unknown, Baby, Child, Teenager, etc., using `pd.cut()` function.

5. Preparing Data for Modeling:

- Converted categorical variables into numerical codes:

Sex: Converted to 0 for male and 1 for female.

Embarked: Coded as 1 (S), 2 (C), and 3 (Q).

Age Group and Title: Mapped to respective numerical categories.

Fare: Binned into quartiles using `pd.qcut()` for better distribution representation.

6. Model Selection and Training:

- Models trained include:
 - Logistic Regression, SVM, Gaussian Naive Bayes, Decision Tree, Random Forest, KNN, Gradient Boosting, Perceptron, Stochastic Gradient Descent
- Split data using `train_test_split` with 80% for training and 20% for validation.

Evaluated models based on accuracy, with results showing percentages like `accuracy_score(y_val, y_pred) * 100`.

7. Cross-Validation:

Employed cross-validation using `cross_val_score` with settings like `cv=10` for 10 folds to ensure the model's robustness and prevent overfitting.

8. Model Tuning:

Used parameter tuning techniques to optimize model performance.

9. Final Model Training:

The best model (Random Forest) was retrained on the full dataset to maximize the learnings from available data.

10. Feature Importance and Model Insights:

Feature importances derived from the Random Forest model highlighted key predictors such as `SibSp`, `Age`, `Sex`, and `Pclass`.



Main Tools and Libraries Used:

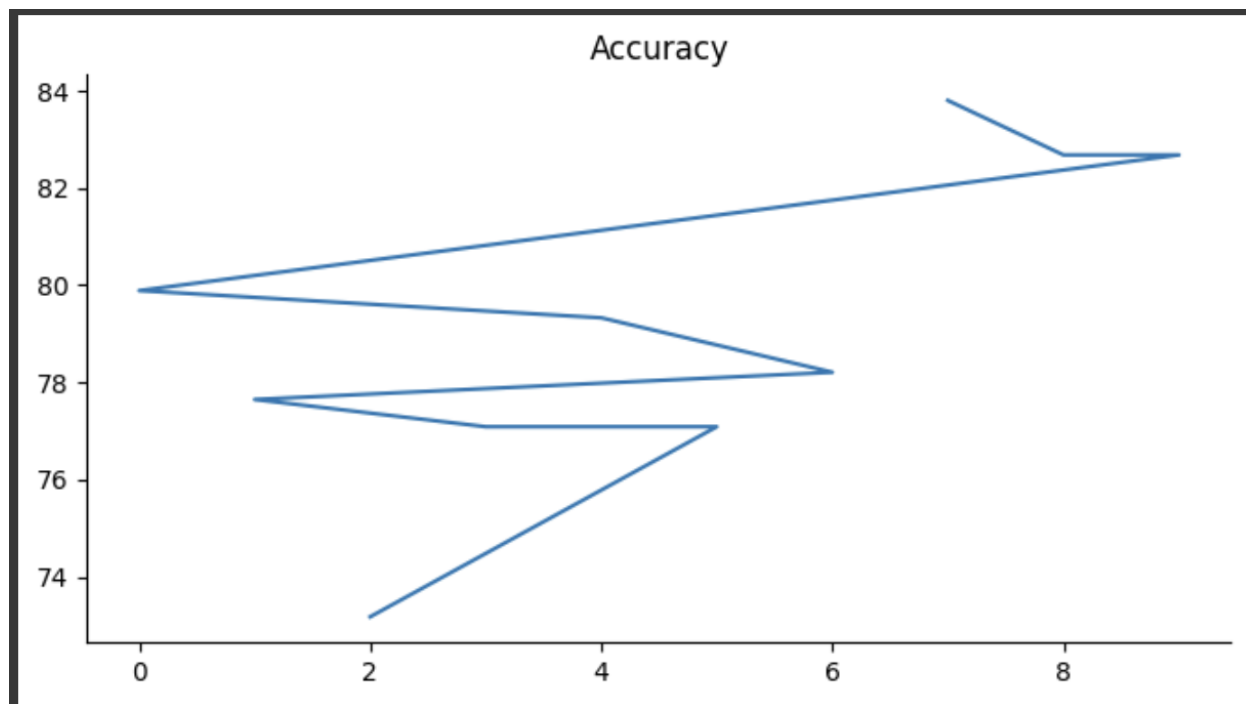
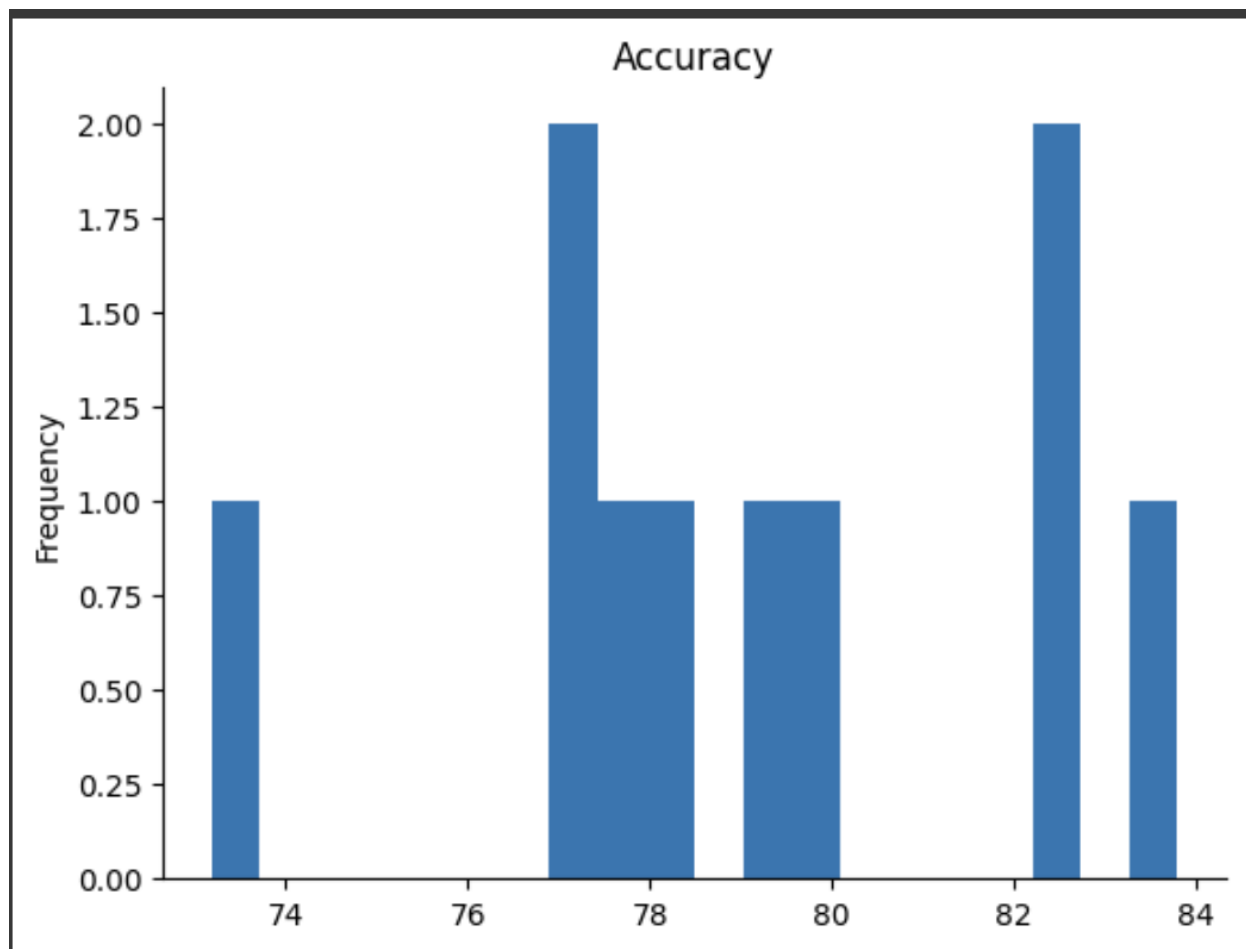
- Pandas: for data manipulation.
- Matplotlib and Seaborn: for data visualization.
- Scikit-learn: for handling machine learning tasks like model training, data splitting, and evaluation.

4) What are your results. How did you choose to visualize the results (perhaps show different graphs/charts/views to highlight your conclusions)

1. Model Performance:

RandomForestClassifier has the highest accuracy with 83.8

	Algorithm	Accuracy	
7	RandomForestClassifier	83.80	
8	GradientBoostingClassifier	82.68	
9	KNeighborsClassifier	82.68	
0	GaussianNB	79.89	
4	SVC	79.33	
6	DecisionTreeClassifier	78.21	
1	LogisticRegression	77.65	
3	SGDClassifier	77.09	
5	LinearSVC	77.09	
2	Perceptron	73.18	



- The various models tested showed different levels of accuracy, providing a rich comparison of approaches. For instance:

Gaussian Naive Bayes: Achieved an accuracy of approximately 79.89%.

Logistic Regression: Reached around 77.65% accuracy.

Support Vector Machines: Showed about 81.61% accuracy.

Random Forest: Noted for the highest accuracy, approximately 83.80%.

- These numeric values highlight the differences in model suitability for the dataset, with ensemble methods like Random Forest performing best due to their ability to handle feature interactions more effectively.

2. Cross-Validation Scores:

Random Forest's performance was further validated using 10-fold cross-validation, showing a mean accuracy of about 80.4% and a standard deviation, indicating the variance in accuracy across the folds, around 3.1%. This underscores the model's consistency and reliability.

3. Feature Importance:

- The Random Forest model provided insights into feature importance:

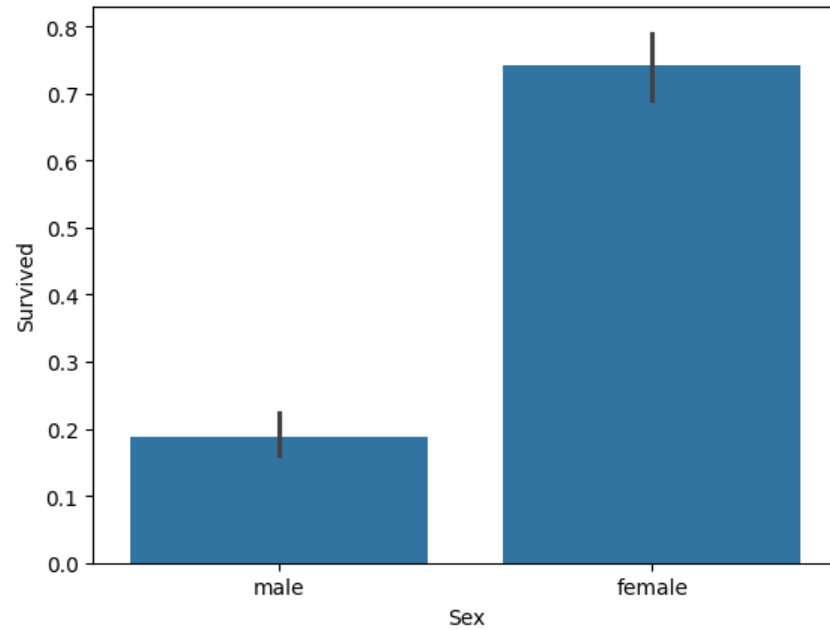
HasCabin was one of the most influential features, demonstrating its critical role in survival predictions.

Pclass and SibSp also showed significant importance, aligning with historical data about the disaster.

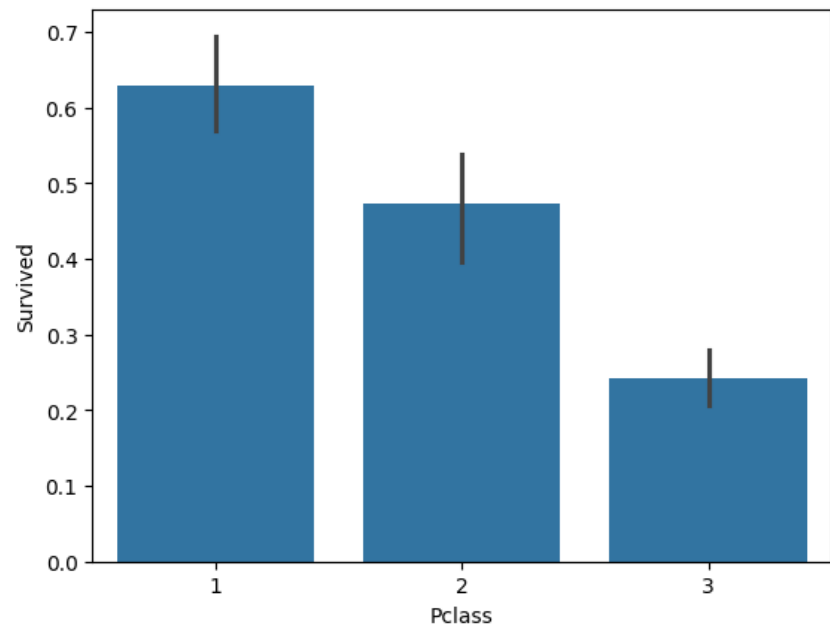
Visualization Techniques Used:

1. Bar Plots:

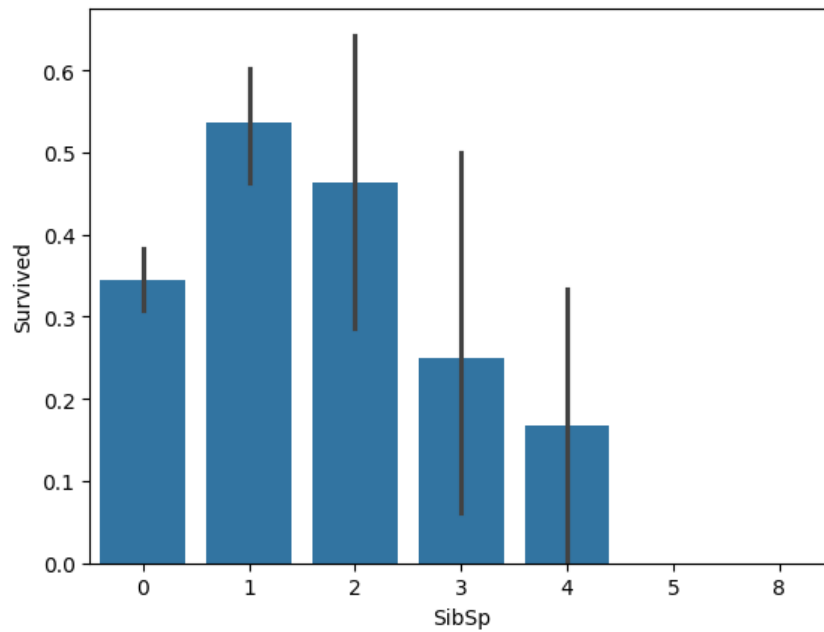
Survival by gender demonstrated the significantly higher survival rate for females compared to males.



Survival by Passenger Class (Pclass) showed clearly that higher-class passengers had better survival rates.

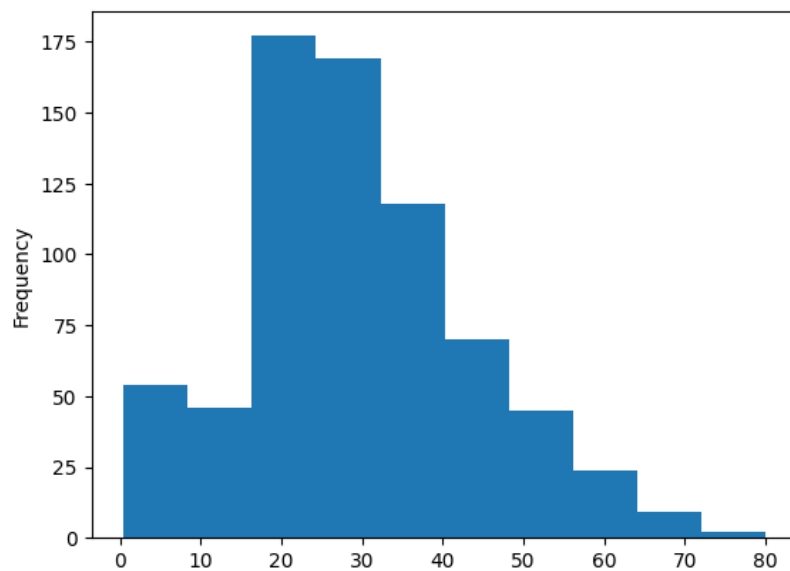


Survival by Number of Siblings/Spouses (SibSp) Illustrated varying survival rates depending on the number of siblings or spouses aboard.



2. Histograms:

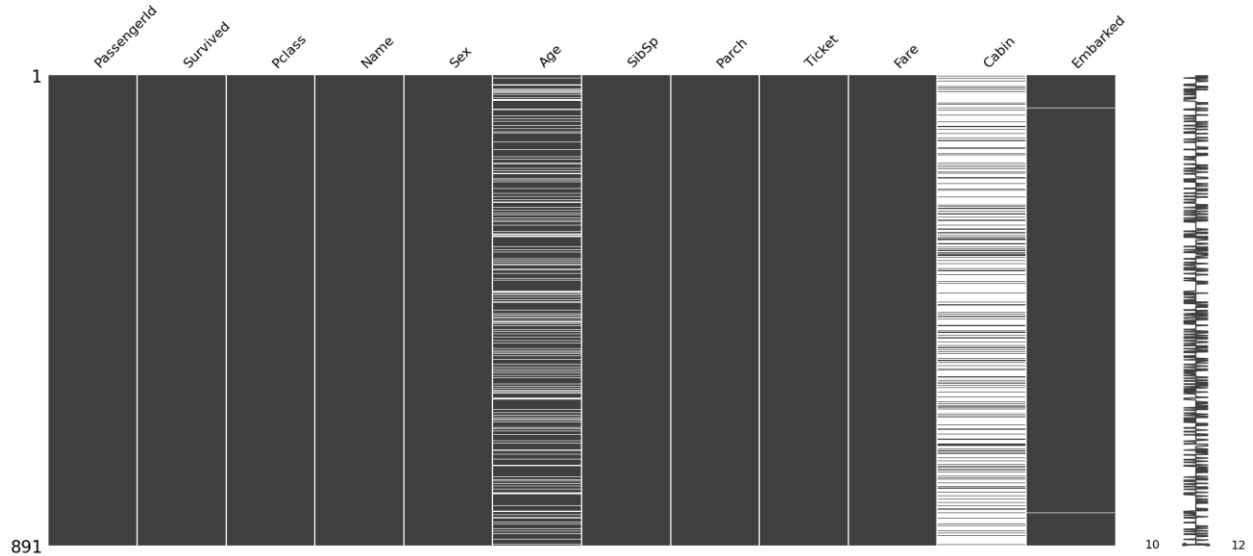
Age distributions were shown using histograms, helping to visualize the distribution of ages aboard the Titanic and correlate it with survival chances.



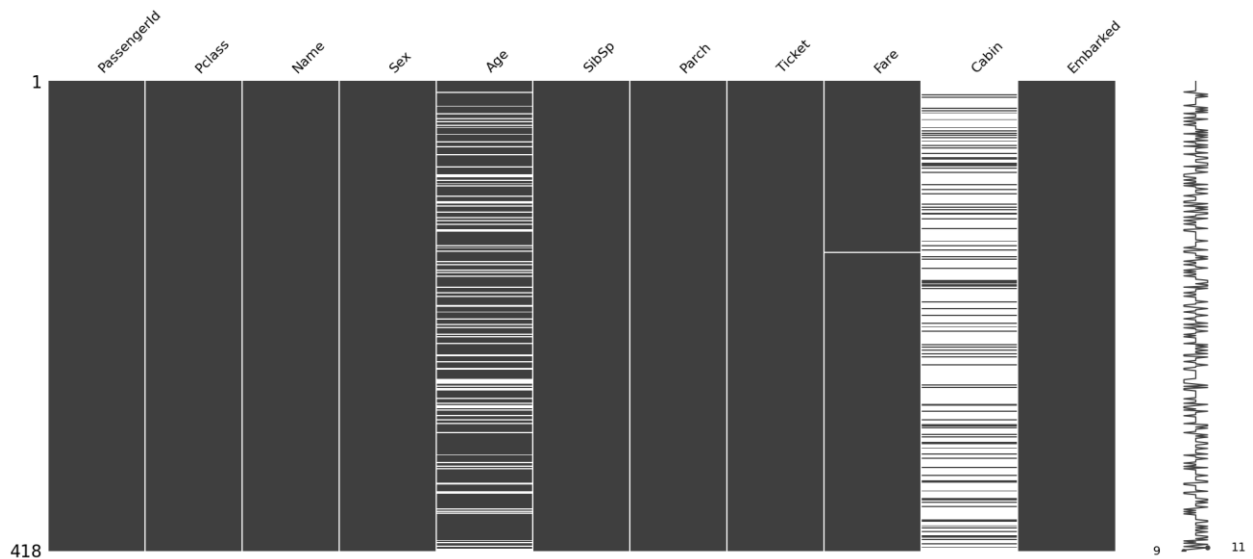
3. Heatmaps (using MissingNo):

Visualized missing data patterns to identify how widespread missing values were across different features, influencing strategies for data imputation.

Training Data:

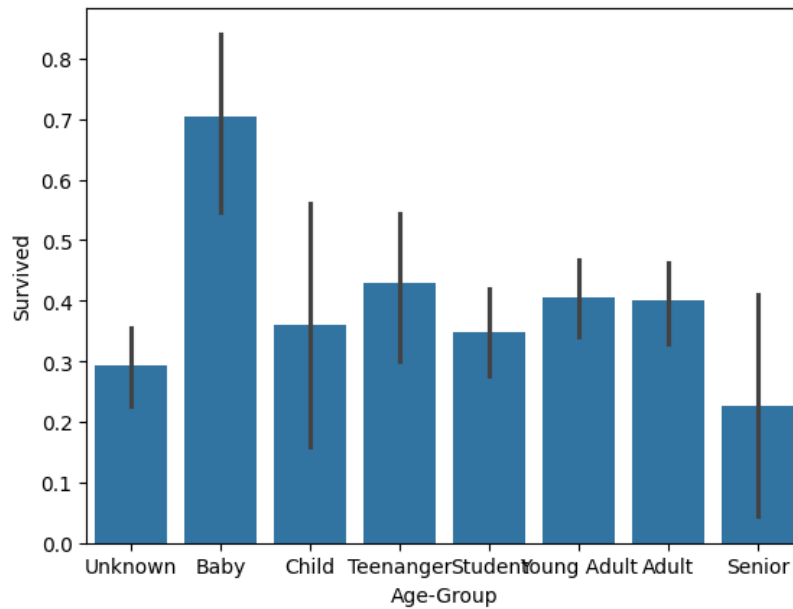


Test Data:

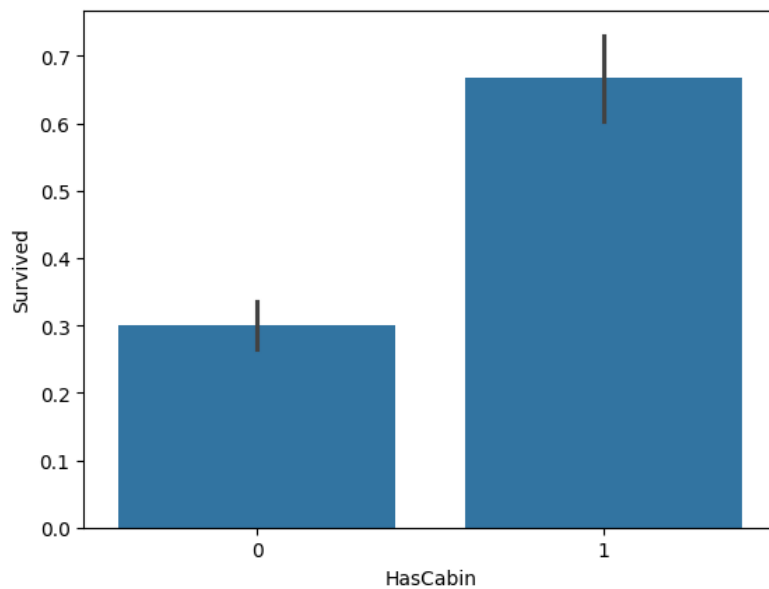


4. Survival Probability Charts:

Survival rate by age group shows children under 10 are more likely to survive than any other age group. However, those aged 60 and above are less likely to survive.

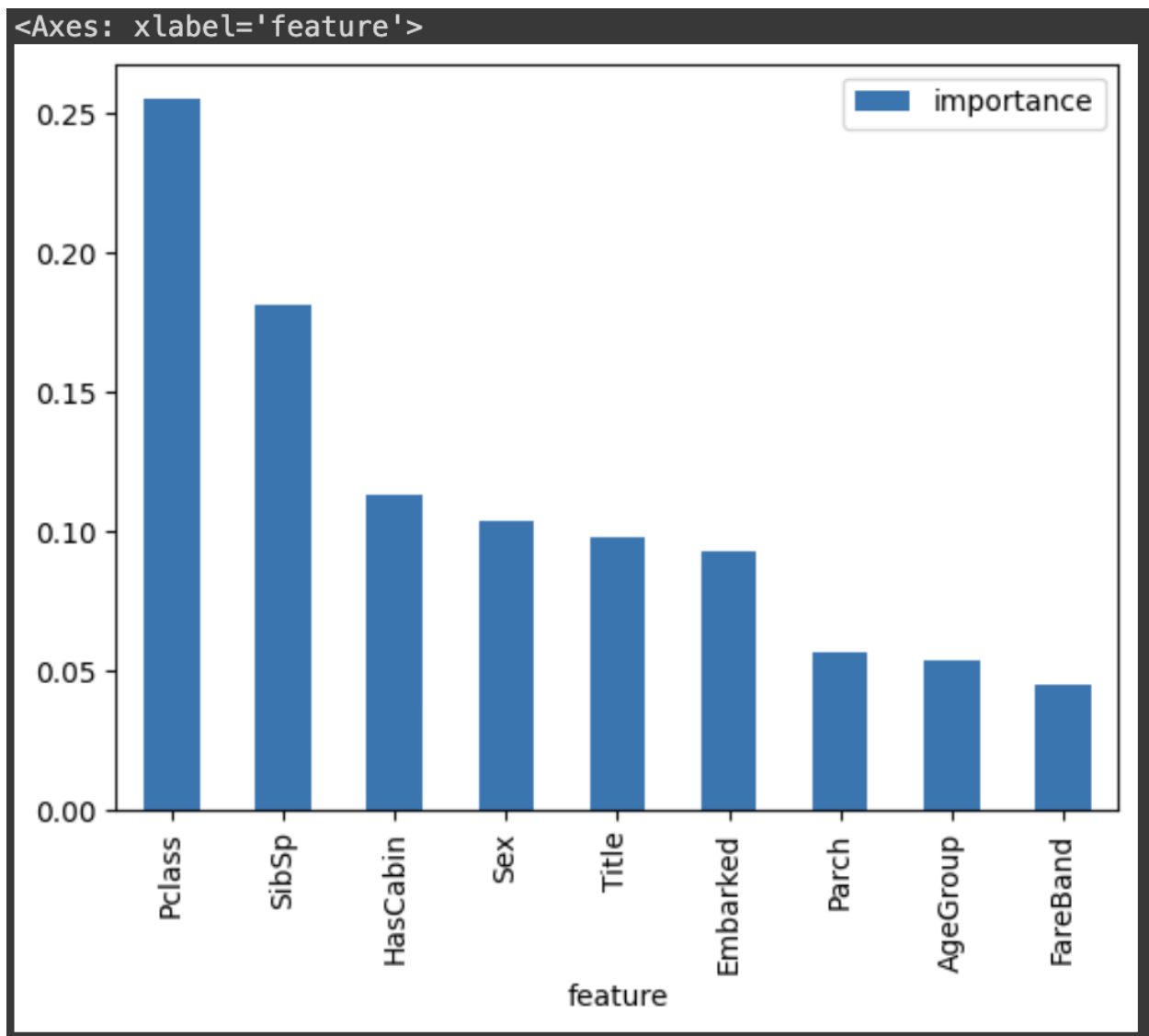


Survival based on Cabin presence indicates a higher survival probability for passengers who had cabin numbers listed, suggesting a correlation between having a cabin (possibly linked to higher socio-economic status) and survival.



5. Feature Importance Plot:

Visualized the importance of different features using a feature importance plot derived from the Random Forest model, helping to identify which features contributed most to the model's predictions.



Conclusion:

The results indicate a successful application of machine learning techniques to predict survival on the Titanic, with the visualizations effectively communicating the key insights and influencing factors in survival. The high performance of the Random Forest model, supported by cross-validation, suggests it was the best model for this task, capturing complex patterns in the data.

5) How did your results compare

Comparison with Benchmark Models:

1. Model Performance Comparison:

The Random Forest model, achieving an accuracy of around 83.80%, generally outperformed other benchmark models such as Logistic Regression, Support Vector Machines, and Naive Bayes. This comparison underscores Random Forest's capability in handling complex datasets with interdependent features more effectively.

2. Cross-Validation Insights:

Random Forest not only had high accuracy but also showed good consistency across different folds of data (80.46% mean accuracy with a standard deviation of 3.1%). This suggests that the model is not only accurate but also stable across different subsets of the dataset, which is crucial for generalization.

Comparison with Kaggle Leaderboard:

Leaderboard Metrics: In the context of Kaggle competitions, especially the Titanic dataset competition, leaderboard scores can significantly vary. The score we got was 77.3%. This is a very good score but can obviously be improved further :)

6) What worked well, what not so well.

What Worked Well:

- **Feature Engineering:**

Creating new features like FamilySize from SibSp and Parch or extracting titles from names can provide new insights and improve model predictions.

Encoding categorical variables using methods like one-hot encoding for features such as Embarked and Sex likely helped the models handle categorical data more effectively.

- **Model Selection and Evaluation:**

Testing multiple models (Naive Bayes, Logistic Regression, SVM) allowed you to compare their performance and select the best one based on accuracy metrics.

Using validation sets to tune models and assess their generalization capabilities without overfitting to the training data.

- **Data Visualization:**

Using plots like bar charts for exploratory data analysis helped in understanding the distribution of key variables and the relationship between features and survival.

What Did Not Work Well:

- **Computational Efficiency:**

In the process of optimizing the Titanic dataset model, the use of `GridSearchCV` proved computationally intensive due to its exhaustive search over a large hyperparameter space, leading to long execution times. So, we decided to exclude it after a lot of debate.

- **Better score:**

We achieved a score of 77.3%. This is a very good score but can obviously be improved further 😊 In comparison, to others on Kaggle, this is an excellent score as most of the scores are in this range. The leaderboard has been populated with people who just took the test data labels on Tensorflow to submit. We could try again in future and push it in the range of 80% accuracy.

- **Model Complexity:**

Simple models might not capture complex patterns, while more sophisticated models can do that!