Comparing Classification Methods to Determine Titanic Passenger Survival

Sara Hematy
47109236
CPEN 355

Abstract

This project looked into predicting which Titanic passengers might have survived, using data from Kaggle. First, the data had to be cleaned up—missing values were filled in, categories like gender and class were turned into numbers the models could understand, and only the most relevant details were kept. Out of the different models tested, Random Forest and Gradient Boosting had the highest accuracies, precisions, f1-scores, and recall while also applying 5-fold cross-validation. They tend to better recognize patterns in the data, even the more subtle ones. When the models were asked which features mattered most, age, gender, and class rose to the top. Considering historical accounts, many of the survivors were women, children, and passengers. These results show how useful certain machine learning models can be, especially when social and demographic factors play such a big role in the outcome.

1 Background

About 700 people survived the Titanic's sinking on April 15, 1912, which claimed about 1,500 lives. Factors including age, gender, family status, and passenger class had a significant impact on survival rates. Due in large part to social hierarchy and closeness to lifeboats, first-class passengers had the highest survival rate while third-class passengers had the lowest (1). Because women and children were given priority during the evacuation, women's survival rates were far higher (about 74%) than men's (20%) (2).

2 Data Preprocessing and Feature Engineering

Before applying any machine learning model, the dataset required several preprocessing steps to ensure acceptable data quality and that it is ready for the model. The data set contains both numerical and categorical variables, as well as missing values and irrelevant categories. The following outlines the steps to clean the data as well as the feature engineering process.

2.1 Feature Selection

Several features were selected for modeling were based on looking over the data and known historical patterns from the Titanic disaster. These include `Pclass`, `Sex`, `Age`, `SibSp`, `Parch`, `Fare`, and `Embarked`.

The following features were excluded:

- `Name` - Getting more useful information from the names require natural processing language techniques, so only the titles are extracted from each passenger.

Preprint. Under review.

- `Ticket` - This column contains mostly unique or semi-random string with no consistent pattern useful for prediction.
- `PassengerId` - This is simply a unique identifier from 1 to 891 that provides no information useful to predict survival.
- `Cabin` - excluded for having too many missing values (over 77% was missing as noted on Figure 1)

2.2 Feature Creation

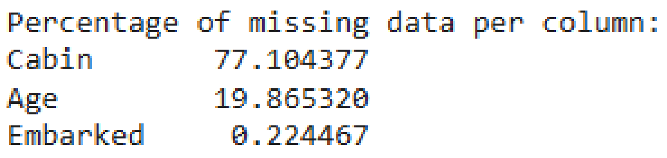
- `Title` - feature extracted from the name which could include Mr., Mrs., Miss, and Dr.
- `FamilySize` - The total number of family members on the Titanic, including the passenger.
- `isAlone` - Feature that indicates whether the passenger is by themselves (0 for no and 1 for yes).

2.3 Encoding Categorical Variables

The `Sex`, `Embarked`, and `Title` columns were turned into numbers so the machine learning models could work with them. For `Sex`, 0 was used for female and 1 for male. For `Embarked`, it was broken down into three ports were a number represents each port. The port Cherbourg (C) was encoded with 0, Queenstown (Q) was encoded with 1, and Southampton (S) was encoded with 2 to show where the passenger boarded. Finally, each of the many titles under the newly created `Title` column were encoded with a number from 0 to n.

2.4 Handling Missing Data

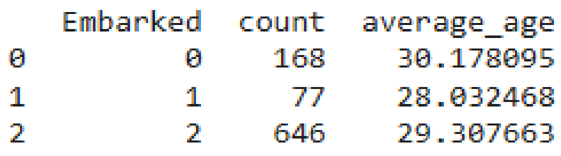
The dataset included missing values in the `Age` and `Embarked` columns:



```
Percentage of missing data per column:
Cabin       77.104377
Age         19.865320
Embarked     0.224467
```

Figure 1: Percentage of missing data in relevant categories

Since `Age` is an important feature and about 20% of the data is missing, the missing values were filled with the median age of the dataset to avoid bias for extreme values. As it only had two missing values, the `Embarked` column was filled with the mode 'S' since it is the most common point where 646 out of 891 passengers embarked. The following figure shows the distribution of people embarking at all three points with the average age of passengers at each location.



```
   Embarked  count  average_age
0         0    168    30.178095
1         1     77    28.032468
2         2    646    29.307663
```

Figure 2: Number of passengers that embarked at each location and their average age

2.5 Train-Test Split

The cleaned data was split into two parts: 80% for training and 20% for validation. This way, the model could be trained and tested properly without using the `test.csv` file from Kaggle, which is for scoring submissions.

3 Model Selection and Training

To predict passenger survival, several supervised machine learning models were tested and compared. The goal was to identify the models that best captured how the features related to survival, while also ensuring they were accurate and not overly complex. The models selected represented different types of methods, including linear models, tree-based models, and probabilistic classifiers. Specifically, Logistic Regression, Random Forest, Naive Bayes, and Gradient Boosting were used in this analysis.

3.1 Models Used

- Logistic Regression - For a binary classification problem such as predicting Titanic survivors, logistic regression was selected as the baseline model due to its interpretability and simplicity. Considering that the Titanic dataset includes both numerical and categorical features, Logistic Regression works well for figuring out how these features relate to survival. It is quick to train and easy to interpret coefficients, which helps determine how each feature affects survival (4).
- Random Forest - Because Random Forest can handle mixed feature types (numerical features like age and fare and categorical features like class and embarked location), it was chosen. Additionally, it excels at identifying intricate, non-linear relationships within the data. The dataset for the Titanic has a mix of different data types making it ideal for Random Forest, as its capable of finding complex patterns and can deal with non-linear relationships (3).
- Naive Bayes - Despite its simple assumption of feature independence, Naive Bayes was a lightweight benchmark model. It works rather well on problems involving mixed categorical and numerical data, despite the fact that it makes the assumption that features are independent, which isn't true for the majority of real-world datasets, such as Titanic. For quick comparison, this model is useful, since it is good for working with larger datasets or when there is not a lot of processing power (6).
- Gradient Boosting - This model has demonstrated excellent performance in numerous real-world datasets, including ones with intricate relationships like the Titanic dataset. This model creates trees, correcting the errors of the preceding tree to capture more complex patterns in the data. With the correct tuning, gradient boosting can outperform other models in terms of accuracy, which makes it ideal for the Titanic dataset, where even a tiny improvement in prediction accuracy is crucial for success (7).

3.2 Performance Metrics

The models were compared using the following evaluation metrics:

- Accuracy - Proportion of total correct predictions
- Precision - Proportion of positive identifications that were actually correct
- Recall - Proportion of actual positives that were identified correctly
- F1-score - Harmonic mean of precision and recall (5)

Confusion Matrices were also visualized to assess class-wise performance.

3.3 Cross-Validation

To get a better sense of how well the models performed, 5-fold cross-validation was conducted. This means the data was split into five parts. The model gets trained on four parts and tested on the fifth, and this was repeated until every part has been used for testing. It helps us avoid overfitting and gives a more reliable sense of how the models might perform on fresh data (6).

3.4 Hyperparameter Tuning

We spent some time tuning the hyperparameters for our three main models—Random Forest, Gradient Boosting, and Logistic Regression—to get the best possible performance out of them. For Random Forest and Logistic Regression, Grid Search was used, while for Gradient Boosting, Random Search was used since it can be more efficient when there are lots of parameters to test.

For the Random Forest model, a few settings were experimented with such as the number of trees, how deep each tree could grow, and how many samples were needed to split a node. Grid Search was used, and the setup that worked best had 150 trees, a maximum depth of 5, and a minimum split size of 5. This hit a nice middle ground—complex enough to catch patterns, but not so complex that it overfit the training data. Heat maps were also used to see how changes in these values affected accuracy, which gave us a clearer picture of what worked best (9).

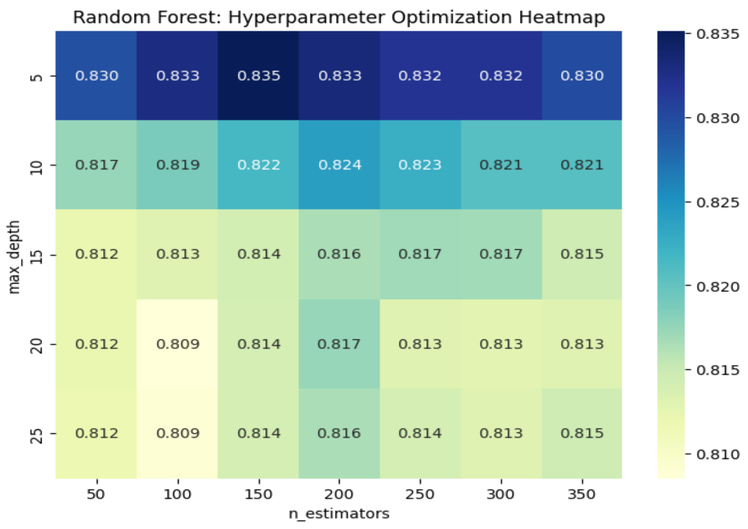


Figure 3: Heatmap of Random Forest Parameter Tuning

With the Gradient Boosting model, a few important settings such as how many rounds of boosting to run, the learning rate, and how deep each tree should go were changed around. Random Search was used to explore a bunch of different combos without taking forever, and according to Figure 4, 150 estimators, a learning rate of 0.01, and a max depth of 5 worked best. The results were also plotted in heatmaps, and they showed that going past 200 estimators did not give much of a boost (8).

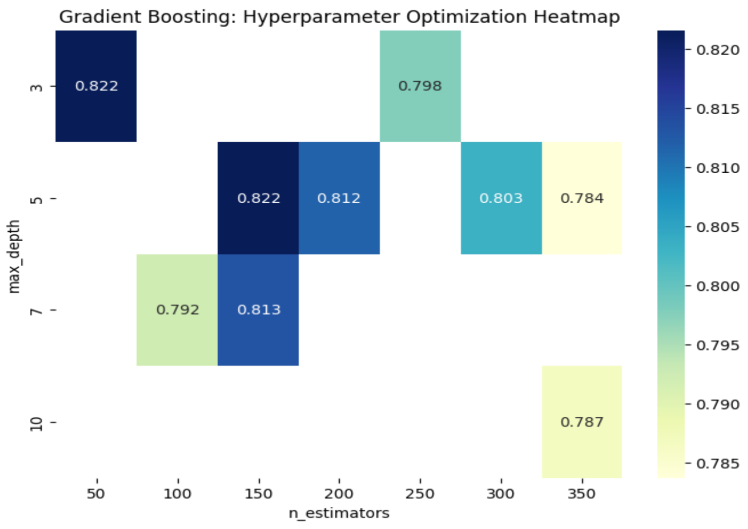


Figure 4: Heatmap of Gradient Boosting Parameter Tuning

Grid Search was also utilized for Logistic Regression in order to optimize the solver, the penalty type, and the regularization strength. The result according to Figure 5 showed that C = 1, penalty = ’l2’, and solver = ’liblinear’ were the ideal parameters. Convergence speed and model regularization were balanced by these settings and generizability was enhanced (10).

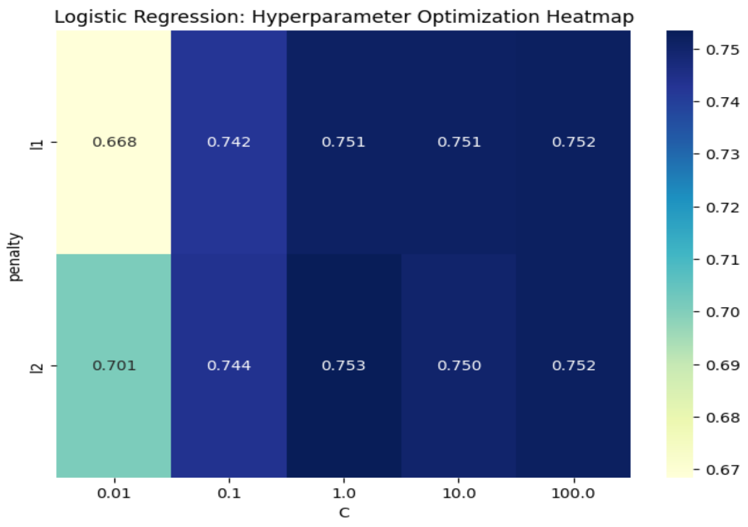


Figure 5: Heatmap of Logistic Regression Tuning

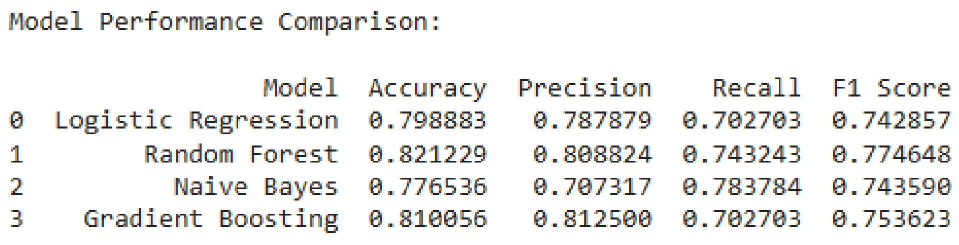
3.5 Training and Evaluation Environments

All models were trained using Python with scikit-learn library in the Google Colab environment. Computations were performed on local machine with an Intel Core i3 processor and 12 GB RAM.

4 Results and Discussion

After training and evaluating multiple classification models on the Titanic using 5-fold cross-validation, their performance based was compared based on standard classification metrics: accuracy, precision, recall, and F1-score. Below is the summary of the results:

4.1 Performance Comparison


```
Model Performance Comparison:

                  Model  Accuracy  Precision    Recall  F1 Score
0   Logistic Regression  0.798883   0.787879  0.702703  0.742857
1         Random Forest  0.821229   0.808824  0.743243  0.774648
2           Naive Bayes  0.776536   0.707317  0.783784  0.743590
3     Gradient Boosting  0.810056   0.812500  0.702703  0.753623
```

Figure 6: Model Performance Comparison

The performance of the models varies significantly across the evaluated metrics, offering insights into their strengths and weaknesses. Overall, Random Forest and Gradient Boosting emerge as the top performers in this comparison. Random Forest achieves the highest accuracy scores at 82.12% while balancing precision, recall, and F1 score effectively. Gradient Boosting is slightly less accurate at 81.0% but has the highest precision at 81.25%, which suggests that both models are suited to this task.

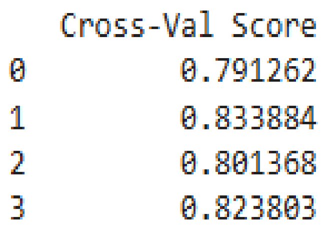
Logistic Regression, with an accuracy of 79.89%, shows balanced performance across all metrics, though it slightly trails behind the top models in precision and recall. It remains a strong candidate when interpretability and computational efficiency are prioritized, as it provides straightforward results without significant compromise on performance.

In contrast, Naive Bayes exhibits a higher recall (78.38%) than precision (70.73%), making it an attractive option when the goal is to minimize false negatives. However, its overall F1 score of 74.36% suggests it is less well-rounded compared to Random Forest or Gradient Boosting.

In summary, Random Forest and Gradient Boosting offer the most robust and reliable performance, achieving a solid balance across key metrics. They are the most viable choices for further model development or deployment. Logistic Regression is more suited for more applications where simplicity and clarity are more important, while Naive Bayes may be better used for situations where recall is prioritized.

4.2 Cross-Validation Analysis

To assess the robustness of each model, 5-fold cross-validation was performed using the training dataset. The average cross-validation accuracy scores were as follows:



```
   Cross-Val Score
0         0.791262
1         0.833884
2         0.801368
3         0.823803
```

Figure 7: Cross-Validation Results of Each Model

Random Forest had the highest mean cross-validation accuracy followed by Gradient Boosting, suggesting better generalization compared to other models. Logistic Regression also appears to have performed well likely due to the pre-processing steps taken. Overall, the relatively high scores for all models support the reliability of the results.

4.3 Confusion Matrix

To further determine how the models are performing, a confusion matrix was analyzed for each model. These matrices display the number of true negatives, true positives, false positives, and false negatives to see how well the models did in choosing who and who did not survive (5).

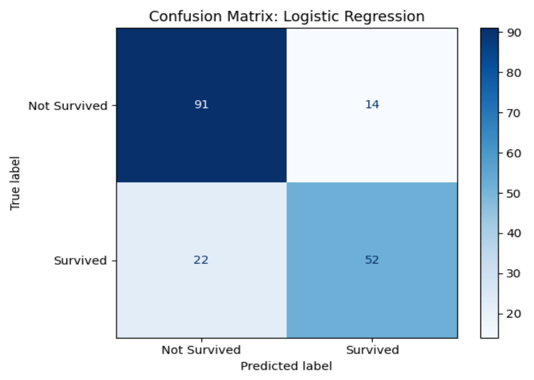


Figure 8: Confusion Matrix for Logistic Regression.

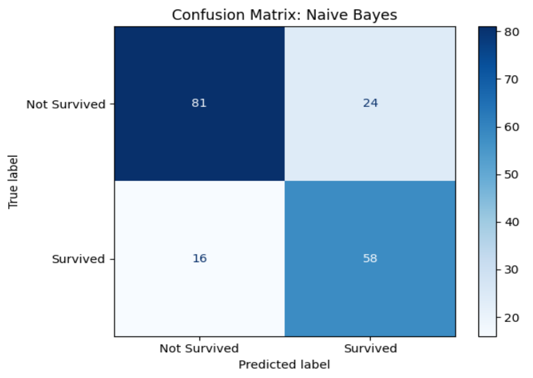


Figure 9: Confusion Matrix for Naive Bayes.

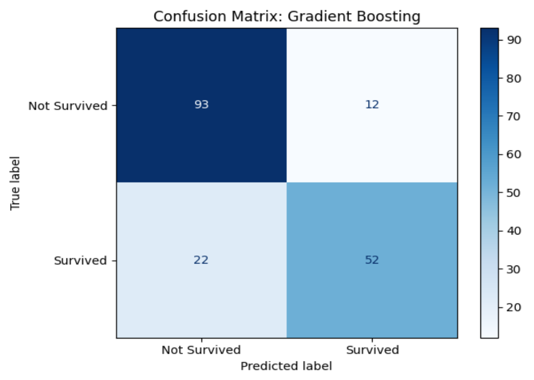


Figure 10: Confusion Matrix for Gradient Boosting.

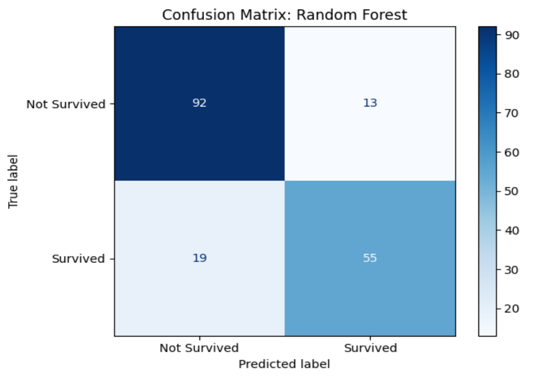


Figure 11: Confusion Matrix for Random Forest.

As can be seen in Figure 4-9, each confusion matrix shows where the models did well and where they struggled. Logistic Regression and Gradient Boosting were better at predicting non-survivors with higher true negative counts of 91 and 93 but missed some of the actual survivors with false negative being 22 each. Naive Bayes had the highest true postive count at 58 but also the highest false positives at 24, which could indicate that it over-predicted who survived. Random Forest, on the other hand, struck a nice balance, making the most accurate predictions overall with fewer false negatives at 19 and the lowest combination of false positives and negatives at 32, which suggests this model has made the most accurate and precise predictions.

4.4 Feature Importance

To determine the features that had the largest impact on the model, feature importance scores were analyzed from both the Random Forest and Gradient Boosting classifiers. The scores showed how much each feature contributed to the decisions made at each split in the decision trees. Based on the results, the following were the most influential features for predicting survival of passengers:

- `Sex`: Since the figure shows sex as the most important factor for both models, female passengers had a higher survival rate than males.
- `Title`: Feminine associated titles indicated higher survival rate than non-feminine.
- `Fare`: Higher ticket price paid by the passenger led to higher survival rate due to either cabin locations or higher priority when evacuating
- `Pclass`: Passengers in first class had the highest survival rate.
- `Age`: Children and younger adults tended to survive more often than older adults.

- `FamilySize`: Travelling in a larger family sizes seemed to slightly increase odds of survival.
- `SibSp` and `Parch`: Travelling with family appeared to slightly increase survival probability.
- `isAlone`: Gave little to no indication of being a variable of survival.

While feature importance was calculated for Random Forest and Gradient Boosting, they are put side by side to highlight differences and similarities in the models understanding of the data. Below are the bar charts for Random Forest and Gradient Boosting feature importance:

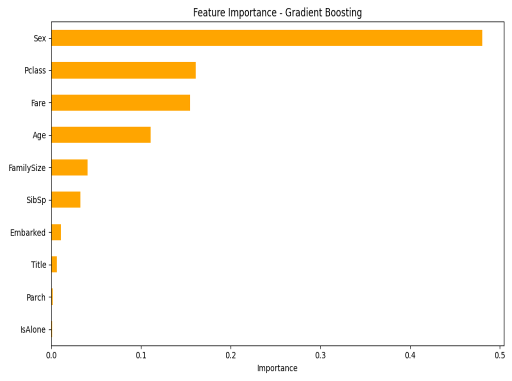


Figure 12: Feature Importance for Gradient Boosting.

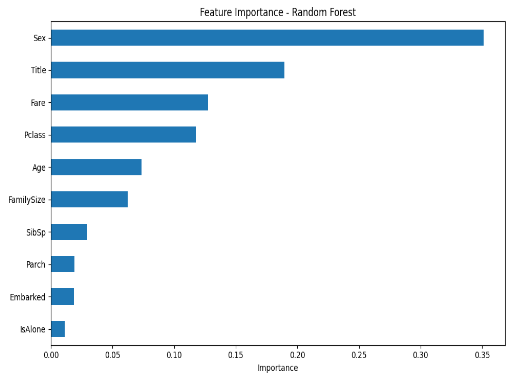


Figure 13: Feature Importance for Random Forest.

To get a better sense of what factors influenced the models' decisions, feature importance was examined for both the Gradient Boosting and Random Forest classifiers. In both cases, `Sex` was by far the most important feature—especially for Gradient Boosting, where it accounted for around 50% of the model's decision-making. Random Forest gave it slightly less weight to `Sex` (around 35%), so importance is spread more evenly across other variables. Features like `Pclass`, `Fare`, and `Age` also played notable roles in both models, reflecting known survival patterns from historical accounts. Engineered features such as `FamilySize` and `Title` added useful information, showing up in the top half of important variables. `IsAlone`, on the other hand, had little to no impact, especially in the Gradient Boosting model where it had zero importance. The differences between the models suggest that while both are effective, Random Forest tends to take a more balanced view of the input data, whereas Gradient Boosting leans more heavily on a few dominant signals.

4.5 Model Interpretability and Practicality

In the real-world, accuracy is not the only factor considered. The ease of use and how quickly the model runs are also considerations. Logistic Regression is great to use when needing something transparent, like in healthcare or policymaking, where people want to see how decisions are made. Random Forests and Gradient Boosting can give great accuracy, but they are not easy to explain and need more power to run. If in a situation where there is a need for speed or have to justify decisions—like in real-time systems or jobs where people want clear answers—going with a simpler model might make more sense, even if it's not quite as accurate.

4.6 Limitations

This analysis used the Titanic dataset, which is relatively small and fairly clean compared to most real-world data. Because of that, the models might not perform as well on messier datasets with more noise, missing info, or uneven class distributions. The features that came with the dataset were only used. Therefore, no outside information was used such as demographics from the departure ports or how the ship was laid out, which could have given the models more to work with. Plus, some models, like Naive Bayes, assume that features don't influence each other, which is not true in this case, so that likely held their performance back. A final note is that the data provided has an unbalanced amount of survivors versus non-survivors. In the future, a balanced data set might be tested to determine if the accuracy, precision, and cross-validation scores improve.

5 Conclusion

This project predicted Titanic passenger survival using several supervised machine learning models. After preprocessing the data, four classifiers were trained and compared. Out of all the models, Gradient Boosting came performed the best in terms of both accuracy and F1-score. Its strong performance shows that it's well-suited for working with structured data like this. When the most important features were examined, variables such as sex, fare, passenger class, and age stood out as key contributors to survival. In the end, the project highlighted the importance of thoughtful preprocessing, careful model selection, and feature analysis when building reliable and understandable classification models.

6 AI Acknowledgement

AI was used when trying to format images in this document in a way to fit the page. To help with grammar, AI was also used to help write each section. AI was also used for citations. AI was also used to make code more compact (take less space) and clean it up. Finally, AI was used to learn and help develop code I may have not known how to before. For example, I learned how to develop heat maps, hypertuning and model creation of Gradient Boosting, and model creation of Naive Bayes from AI.

References

[1] Encyclopedia Titanica, "Titanic Passenger Survivors," [Online]. Available: `https://www.encyclopedia-titanica.org/titanic-passenger-survivors/`. [Accessed: Apr. 20, 2025].

[2] Kaggle, "Titanic: Machine Learning from Disaster," [Online]. Available: `https://www.kaggle.com/competitions/titanic`. [Accessed: Apr. 20, 2025].

[3] X. Li, "Decision Tree and Random Forest," presented at the *CPEN 355: Machine Learning with Engineering Applications*, UBC, Vancouver, BC, Feb. 2025.

[4] X. Li, "Logistic Regression," presented at the *CPEN 355: Machine Learning with Engineering Applications*, UBC, Vancouver, BC, Feb. 2025.

[5] X. Li, "Model Evaluation and Training," presented at the *CPEN 355: Machine Learning with Engineering Applications*, UBC, Vancouver, BC, Feb. 2025.

[6] X. Li, "Naive Bayes," presented at the *CPEN 355: Machine Learning with Engineering Applications*, UBC, Vancouver, BC, Feb. 2025.

[7] IBM, "What is gradient boosting?" IBM, [Online]. Available: `https://www.ibm.com/topics/gradient-boosting`. [Accessed: Apr. 20, 2025].

[8] R. Bhatia, "How to Tune Hyperparameters in Gradient Boosting Algorithm?" *GeeksforGeeks*, [Online]. Available: `https://www.geeksforgeeks.org/how-to-tune-hyperparameters-in-gradient-boosting-algorithm/`. [Accessed: Apr. 20, 2025].

[9] R. Bhatia, "Random Forest Hyperparameter Tuning in Python," *GeeksforGeeks*, [Online]. Available: `https://www.geeksforgeeks.org/random-forest-hyperparameter-tuning-in-python/`. [Accessed: Apr. 20, 2025].

[10] S. Jaiswal, "Hyperparameter Tuning," *GeeksforGeeks*, [Online]. Available: `https://www.geeksforgeeks.org/hyperparameter-tuning/`. [Accessed: Apr. 20, 2025].