COMP4651 Cloud Computing

Project Report
Group 13 - Topic 2

Conducted by
Chan Kwan Ho, LAM Coco Wai Kwan, YEN Yiu Wai
and YUEN Man Him

Overview

The sinking of the RMS Titanic on April 15, 1912, led to the heartbreaking loss of 1,502 individuals out of a total of 2,224 passengers and crew members, following its collision with an iceberg. This project seeks to develop a predictive model aimed at identifying the traits of passengers who had a higher likelihood of surviving this disaster. It will examine a range of factors, including age, gender, socio-economic status, and familial connections. Through the application of various classification models within a PySpark framework, we utilized machine learning (ML) methodologies to reveal patterns and insights that elucidate the determinants of survival during this significant historical incident.

Data visualization

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen ...	male	22.0	1	0	A/5 21171	7.25	NULL	S
2	1	1	Cumings, Mrs. Joh...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. ...	female	26.0	0	0	STON/O2. 3101282	7.925	NULL	S

Figure 1. Head of Training Data

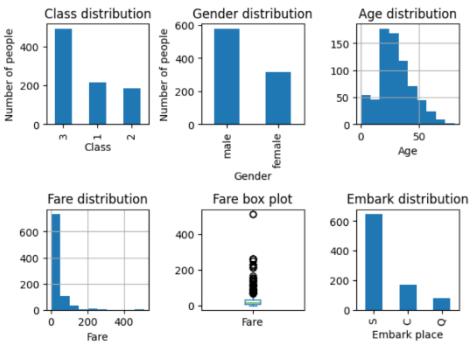


Figure 2. Data Visualization

Data visualization is essential for uncovering patterns within the Titanic dataset, enabling the presentation of various significant insights through multiple visual formats. A bar chart depicted the passenger distribution, illustrating a predominance of males and a significant concentration in the third class, thereby highlighting socio-economic inequalities. Furthermore, the age distribution indicated that most passengers fell within the 20 to 40 age brackets, while a box plot represents the distribution of ticket fares, suggesting a relatively uniform fare structure across the dataset. Lastly, a bar chart was utilized to represent the embarkation ports of the passengers. Collectively, these visualizations deepened our comprehension of the factors that influenced survival during the Titanic tragedy.

Pyspark environment

In this project, we employed the PySpark framework to take advantage of its distributed computing capabilities for the efficient processing of extensive datasets. PySpark significantly enhances performance by providing scalability, which enables us to manage large volumes of data across various nodes, a critical factor as our dataset expands. Its ability to execute tasks in parallel accelerates processing speed, which is particularly beneficial for iterative

algorithms and ML applications. Furthermore, PySpark guarantees fault tolerance through the use of Resilient Distributed Datasets (RDDs). With its intuitive API and strong support for ML, PySpark represents an optimal solution for our project, allowing us to extract meaningful insights from the Titanic dataset.

Modeling

In the following, three different models will be presented and evaluated. They are:

1. Extreme Gradient Boosting (XGBoost)
2. Random Forest (RF)
3. Ensemble of Naïve Bayes (NB), Linear Support Vector Machine (LinearSVC), and Multilayer Perceptron (MLP) Classifier

Extreme Gradient Boosting (XGBoost)

XGBoost is a powerful and versatile ensemble learning technique that builds upon the principles of gradient boosting. It constructs a series of decision trees in a sequential manner, where each new tree corrects the errors made by the previously trained trees. This method is particularly effective for both classification and regression tasks, making it a favored choice in data science competitions and real-world applications.

One of the key strengths of XGBoost is its ability to handle missing values automatically, which simplifies data preprocessing. The algorithm also includes built-in regularization techniques (L1 and L2), which help to mitigate overfitting by penalizing overly complex models. This is particularly useful in high-dimensional datasets like the Titanic dataset, where many features can contribute to noise.

A notable feature of XGBoost is its parallel processing capability. By utilizing multiple CPU cores, it significantly speeds up the training process, making it suitable for large datasets. Additionally, XGBoost supports various objective functions and evaluation metrics, allowing it to be tailored to specific problems.

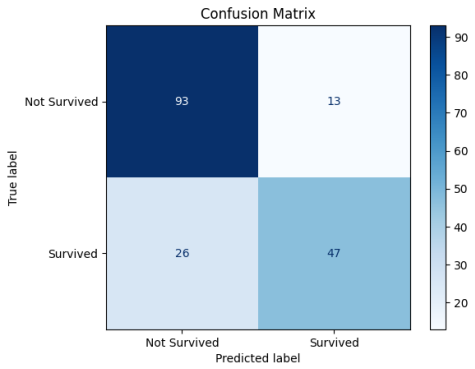


Figure 3. XGBoost's Confusion matrix

In practice, XGBoost has demonstrated impressive performance in binary classification tasks, such as predicting survival on the Titanic. Its ability to capture complex patterns through decision trees, combined with its efficiency and flexibility, positions it as a robust option for tackling various ML challenges.

In our analysis, the implementation of the XGBoost algorithm achieved a commendable accuracy rate of 78.21% in classifying survival outcomes. This level of accuracy suggests that the model effectively captured the underlying patterns in the data, distinguishing between survivors and non-survivors with notable precision.

To enhance the model's performance, we addressed missing values through imputation techniques. Specifically, we filled missing values in the "Age" column with the median age, and we used the most frequent value for the "Embarked" column. This approach allowed us to retain as much valuable data as possible, minimizing information loss. Additionally, we encoded categorical features into numerical representations using StringIndexer and one-hot encoding. This transformation was crucial, as it enabled the XGBoost model to process the data effectively and learn from the training dataset, optimizing its parameters to achieve high accuracy. We evaluated the model's performance on the training set, where it demonstrated an impressive accuracy of 78.21%. This result underscores the model's ability to generalize well to unseen data, indicating its robustness and reliability in predicting survival outcomes.

Random Forest (RF)

Random Forest is an influential ensemble learning technique that generates numerous decision trees during the training phase and determines the most frequent class among the individual trees for classification tasks. This approach is especially beneficial for binary classification problems (i.e. the titanic survive problem), as it effectively manages extensive datasets, accommodates high dimensionality, and reduces the risk of overfitting through averaging. A notable advantage of employing RF within the PySpark framework is its ability to utilize distributed computing. By harnessing RDD, PySpark facilitates the efficient handling of large data volumes across multiple nodes in a cluster, leading to quicker execution times and enhanced scalability. Consequently, RF emerges as a highly suitable option for big data applications in binary classification contexts.

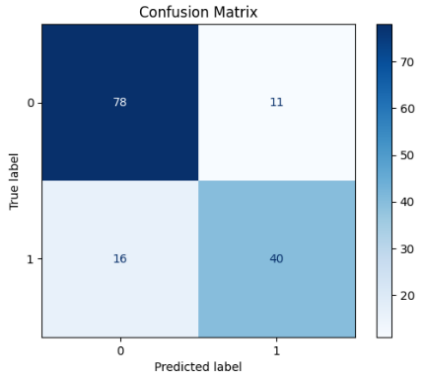


Figure 4. RF's Confusion Matrix

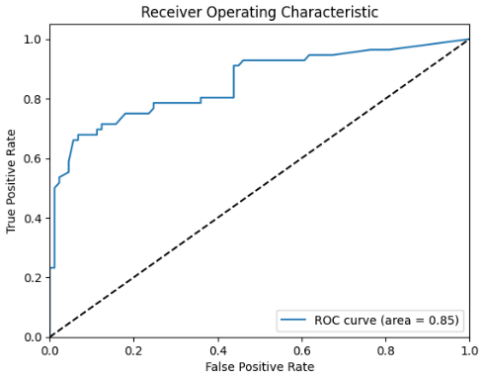


Figure 4. RF's AUC Graph

In our analysis, the foundational RF demonstrated a respectable accuracy rate of 81% in the survival classification. Subsequently, we engaged in hyperparameter tuning, examining

various configurations to refine the model's efficacy. Furthermore, we applied ensemble learning strategies, integrating Gradient Boosted Trees (GBT) to augment our predictive capabilities. Nevertheless, despite these enhancements, we did not detect a notable increase in accuracy beyond the initial 81%. This observation indicates that the RF was already operating at a commendable level, and the complexities introduced by hyperparameter adjustments and ensemble techniques did not provide significant advantages in this specific instance.

One viable approach to enhancing the precision of our model is to augment the quantity of training data. An increased dataset can furnish the model with a broader array of examples, thereby facilitating more effective learning and improved generalization to novel instances. This can be accomplished through techniques such as data augmentation, as acquiring additional labeled data from external sources may not be feasible in the Titanic situation.

In summary, although RF has demonstrated its efficacy as a binary classification algorithm, especially within a distributed computing framework like PySpark, there is still room for further refinement. By concentrating on expanding the dataset and enhancing feature engineering practices, we can aim for superior performance in subsequent iterations of our model.

Ensemble of NB, LinearSVC, and MLP

Naïve Bayes, Linear Support Vector Machine (LinearSVC), and Multilayer Perceptron Classifier (MLP) are commonly known as the simplest algorithms in ML:

A parallel ensemble approach, an ML technique that aggregates two or more learners, is adopted to combine these three models in order to achieve better accuracy. In the model, we make use of each model's f1 score to perform a weighted sum on the prediction column so to produce the final prediction.

On the other hand, while maintaining good accuracy, it is also important to maintain a high predictive of the model, which the following will evaluate.

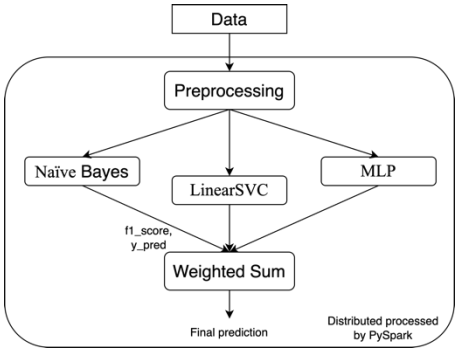


Figure 5. Ensemble model framework.

Split ratio	[0.7, 0.3]	[0.6, 0.4]	[0.5, 0.5]	[0.4, 0.6]	[0.3, 0.7]
Epoch	100	500	1000	1500	1800
Layers	[7, 4, 2]	[7, 32, 64, 16, 2]			
Block size	16	64	128	256	

Table 1. Set of hyperparameters and train test split ratio

As we need to tune a lot of hyperparameters and test different train-test splits for evaluating the predictiveness of the model, PySpark is adopted to speed up the tuning process by its distributed processing

Result:

Epoch	Layer	Block size
1500	[7, 32, 64, 16, 2]	64

Table 2. Best parameter when splitting half

	Precision	Recall	F1-score	Support
0	0.83	0.89	0.86	276
1	0.8	0.7	0.75	169
Accuracy			0.82	445
Macro avg	0.81	0.8	0.8	445
Weighted avg	0.82	0.82	0.82	445

Table 3. Classification report of ensemble

$Accurracy = 0.8202247191011236$

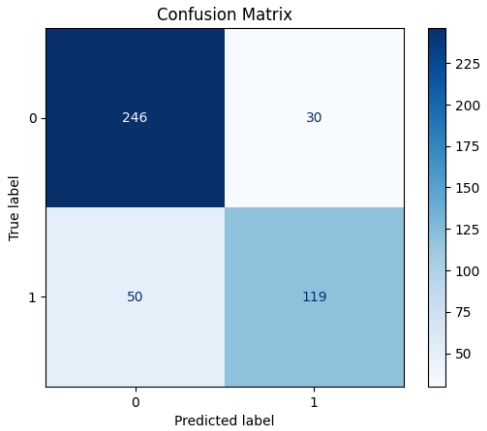


Figure 6. Ensemble's confusion matrix

In comparison to the predictiveness of the model, the graph shows different accuracies in different splitting, where 50:50 has the highest accuracy. Providing more data to this model higher the chance of overfitting it. Less training data makes it more predictive and has a higher chance of performing better in the actual testing data set.

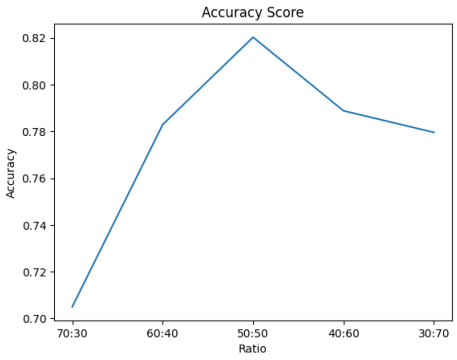


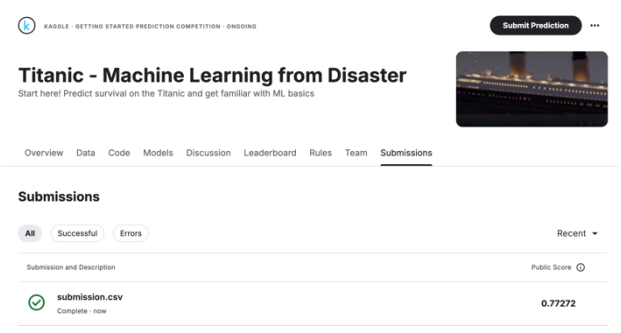
Figure 7. Comparison of Accuracy

In evaluating the difference between using PySpark and ordinary MLP, we build another model using TensorFlow MLP. Here is the time needed per iteration in tuning hyperparameters.

PySpark MLP	TensorFlow
33m 15.7s	26m 13.7s

We found that using PySpark did not improve efficiency as expected. We believe this is due to the size of the data; PySpark typically performs better with larger datasets. However, on our local machine, which operates as a single node, processing large datasets may not be efficient or convenient. In our current scenario, it is sufficient to complete this task well.

Conclusion



This project successfully developed predictive models to identify the traits of Titanic passengers that influenced survival outcomes. By utilizing machine learning techniques such as Extreme Gradient Boosting, Random Forest, and an ensemble approach, we achieved notable accuracy rates, with the ensemble approach performing the best at 82% in validation, 77.272% in test data. The analysis also revealed the performance of PySpark when using a small data set.