python-docx

Ned Palom  Kaggle Competition

Titanic – Machine learning from disaster

# Table of Contents

1. Introduction

2. Dataset Description

3. Exploratory Data Analysis (EDA)

4. Data Preprocessing

5. Modeling

6. Results

7. Conclusion

8. Future Work

9. Appendix

# 1. Introduction

**Problem Statement:**  
The project addresses the problem of predicting the survival of passengers aboard the Titanic based on various features such as age, sex, class, and ticket information. This problem is framed as a binary classification task, where the goal is to determine whether a passenger survived or not based on historical data (https://www.kaggle.com/competitions/titanic)

**Objective:**  
The objective of the project is to build a machine learning model that accurately predicts the survival of passengers using the Titanic dataset. By analyzing the dataset, the project also aims to understand the key factors that influenced survival rates during the disaster

# 2. Dataset Description

The Titanic dataset contains information on 891 passengers, including features such as:

* **Survival** (0 = No, 1 = Yes)
* **Pclass** (Passenger class: 1st, 2nd, 3rd)
* **Sex**
* **Age**
* **SibSp** (Number of siblings/spouses aboard)
* **Parch** (Number of parents/children aboard)
* **Ticket** (Ticket number)
* **Fare** (Ticket fare)
* **Cabin** (Cabin number, with many missing values)
* **Embarked** (Port of embarkation: C = Cherbourg, Q = Queenstown, S = Southampton)

- **Number of rows and columns.**In the Titanic dataset, there are a total of **11** columns and **891** rows

- **Missing Values**

There are **868** missing values in the Train dataset, and **413** missing values in the Test dataset

**Technologies Used**

* **Programming Language:** Python
* **Libraries and Tools:**
  + **Pandas** and **NumPy** for data manipulation and analysis
  + **Scikit-learn** for building machine learning models
  + **Jupyter Notebook** for coding and experimentation​

# 3. Exploratory Data Analysis (EDA)

**Summary statistics (mean, median, mode, etc.)**

|  |  |
| --- | --- |
| Train Dataset | |
| **Columns** | **Central Tendency Matrix (mean, mode, median)** |
| Cabin | Mode |
| Embark | Mode |
| Age | Mean |
| Test Dataset | |
| Cabin | Mode |
| Embark | Mode |
| Fare | Mean |
| Age | Mean |

Note: *For the ‘Age’ column I filled all the missing values with the mean function and rename the column into ‘Column\_Age’ for both datasets.*

# 4. Data Preprocessing

To prepare the dataset for modeling, the following steps were taken:

1. **Handling Missing Values**

Missing values in the dataset were addressed to ensure no null values would affect the modelling process.

* **Cabin**: Missing values in the Cabin column were replaced with the most common value (mode) in that column.
* **Embarked**: Missing values in the Embarked column were replaced with the most frequent value (mode).
* **Age**: A new column, Column\_Age, was created to handle missing values in the Age column. Missing values were replaced with the mean of the column.
* **Fare**: Missing values in the *Fare* column were replaced with the mean in the ‘**Test Dataset’**

Afterward, the original Age column was updated with the filled values, and Column\_Age was dropped to avoid redundancy.

1. **Feature Engineering**

Feature engineering was limited but included replacing missing values, creating new columns, and ensuring data consistency.

* **Column Creation**: Column\_Age was created to handle missing Age values, ensuring no nulls were present in this key feature.
* **Redundant Features**: The Age column was dropped after Column\_Age was created.

1. **Encoding Categorical Variables**

Categorical variables in the dataset (e.g., Embarked, Sex, Cabin) need to be encoded into numeric values for machine learning models. Though not explicitly shown in the code, typical encoding strategies include:

* **Label Encoding:** Assigning a numeric label to each category.
* **One-Hot Encoding**: Converting each category into a binary column.

1. **Data Normalization or Scaling**

For some models (e.g., logistic regression or neural networks), numerical features need to be scaled to improve model convergence and performance. While this step is not shown in the provided code, the following steps would be recommended:

* **Scaling Using StandardScaler**:

# 5. Modeling

The model used in this code is **Logistic Regression**. It is a linear model commonly used for binary classification problems, like predicting survival in this dataset.

**Model Training**

The data preparation and splitting process were as follows:

1. The target column, 'Survived', was specified as the dependent variable.
2. The remaining columns were treated as features (independent variables).
3. The dataset was split into **training (80%)** and **testing (20%)** sets using the train\_test\_split() function from sklearn, with a random\_state=42 to ensure reproducibility.

**Hyperparameter Tuning**

No hyperparameter tuning was performed in this code. The **default parameters** of the LogisticRegression model were used.

**Evaluation Metrics**

The following metrics were used to evaluate the model:

1. **Accuracy Score:** This measures the proportion of correctly classified samples in the test set.
   * Computed using accuracy\_score.
2. **Confusion Matrix:** This provides detailed counts of true positives, true negatives, false positives, and false negatives.
   * Computed using confusion\_matrix.
3. **Classification Report:** This includes precision, recall, F1-score, and support for each class.
   * Generated using classification\_report.

**Output:**

After running the code, the following results will be printed:

1. **Accuracy**: A single numeric value indicating how well the model performs overall.
2. **Confusion Matrix**: A table showing the breakdown of correct and incorrect predictions.
3. **Classification Report**: A detailed breakdown of performance metrics (precision, recall, F1-score) for each class.

Additionally, the trained Logistic Regression model is saved to a file named model\_1 using the joblib library.

# 6. Conclusion

**Findings and Overall Performance:**

The machine learning model developed to predict the survival of passengers from the Titanic disaster has shown promising results. By leveraging the data from the dataset, the model effectively identifies patterns associated with survival. The model achieved an accuracy of approximately **79%** and a precision/recall score of **0.83**, indicating a satisfactory performance in distinguishing between survivors and non-survivors.

The findings align with historical insights, such as the higher survival rates of women and children and passengers in higher-class cabins. This demonstrates the model's ability to capture

**Strengths:**

1. **Feature Importance:** The model successfully utilized significant features like gender and passenger class, which had strong correlations with survival.
2. **Model Simplicity:** The choice of algorithm (Logistic Regression) provided interpretability while achieving reasonable performance.
3. **Real-World Applicability:** The model demonstrates how machine learning can analyse historical data to uncover meaningful patterns and predictions.

**Limitations:**

1. **Data Imbalance:** The dataset may have imbalanced classes, with fewer survivors compared to non-survivors, potentially affecting the model’s ability to generalize.
2. **Missing Data:** Certain features, such as **Age**, **Cabin**, **Embark**, and **Fare** had missing values that required imputation, which could introduce bias.
3. **Feature Engineering:** While the current features provided good predictive power, additional domain-specific features (ticket price in relation to wealth) were not fully explored.
4. **Overfitting/Underfitting:** Depending on the complexity of the chosen algorithm, the model might suffer from overfitting or underfitting, limiting its robustness on unseen data.

**Potential Improvements:**

1. **Feature Enrichment:** Incorporate additional features, such as the port of embarkation, ticket price, or family relationships, to improve the model’s predictive capability.
2. **Advanced Models:** Experiment with ensemble methods (Gradient Boosting, XGBoost) or deep learning techniques to improve performance.
3. **Hyperparameter Tuning:** Perform systematic hyperparameter optimization (e.g., GridSearch or RandomizedSearch) to fine-tune the model.
4. **Cross-Validation:** Use cross-validation techniques to ensure the model generalizes well to unseen data.
5. **Handling Missing Data:** Employ more sophisticated techniques for imputing missing data, such as predictive imputation using other features.
6. **Address Data Imbalance:** Use techniques such as oversampling, undersampling, or SMOTE to address class imbalance and improve the model’s ability to predict survivors accurately.

By addressing these limitations and potential improvements, the model can be further refined to provide even more accurate and reliable predictions. Overall, the project showcases how machine learning can extract insights from historical data and create predictive systems with real-world relevance.

real-world trends and validate its predictive power.

# 7. Future Work

To extend the project where I developed a machine learning model using the LogisticRegression method to predict Titanic survivors based on a Kaggle dataset, future work could include the following opportunities:

1. **Implementing Neural Networks**  
   Neural networks, such as deep learning models, could be employed to improve prediction accuracy by capturing complex patterns and relationships within the dataset. This approach may involve experimenting with architectures like feedforward neural networks, recurrent neural networks (RNNs), or convolutional neural networks (CNNs), depending on the dataset's features.
2. **Deploying the Model as a Web Application or API**  
   The logistic regression model could be deployed as a web application or RESTful API, allowing users to input passenger details (e.g., age, gender, class) and receive survival predictions. Frameworks like Flask, FastAPI, or Django could be used for deployment. This would enhance the accessibility and usability of the model for end users.

# 9. Appendix

Github: (https://github.com/NedPalom/AI-and-Cloud-Projects)

**Code**: The code can be found at the link below:

https://colab.research.google.com/drive/1KHUcIPyTtqFyj\_spKBpJlhqxBhaDYx5-#scrollTo=htmK1BZpb6nv

**References:**

* + Assistance from ChatGPT
  + Instructions and dataset from Kaggle (Titanic – Machine Learning from Disaster)