Universidad Autónoma de Querétaro
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Faculty of Engineering
MCIA

First Term Exam

K Nearest Neighbors (KNN)

Student:
Juan Manuel
Aviña Muñoz

Professor:
Dr. Marco
Aceves Fernandez

August 20th, 2023

Contents

1 Objectives

The primary aim of this project is to implement a K-Nearest Neighbors (KNN) classifier to predict survival probabilities using a dataset related to the tragic sinking of the RMS Titanic in 1912. The dataset contains information about the passengers, including features such as age, gender, class, and a binary indicator of whether they survived or not. Our goal is to develop a KNN model that can effectively predict the likelihood of survival based on these features.

2 Introduction

The sinking of the RMS Titanic in 1912 is one of the most infamous maritime disasters in history. In this activity, we embark on a data-driven journey to explore the fate of Titanic passengers and predict their survival probabilities using a machine learning approach, specifically the K-Nearest Neighbors (KNN) algorithm.

KNN, a straightforward yet powerful machine learning algorithm, operates on the principle that similar data points tend to have similar outcomes. It accomplishes this by looking at the ”K” nearest neighbors of a given data point and making predictions based on the majority class among those neighbors. We will implement KNN to classify passengers as survivors or non-survivors, using features such as age, gender, and class as factors influencing their chances of survival.

3 Theoretical Foundation

K-Nearest Neighbors (KNN) is a popular and simple machine learning algorithm used for both classification and regression tasks. It is a type of instance-based learning, or lazy learning, where the algorithm makes predictions based on the similarity of input data points to the data it was trained on. KNN is non-parametric, meaning it doesn't make any underlying assumptions about the data distribution.

KNN is easy to understand and implement, making it a good choice for simple tasks or as a baseline model. However, it has some limitations. Here's an overview of how the algorithm works:

1. **Initialization:** Start with a dataset containing labeled examples, where each example has a set of features and a corresponding class label (for classification) or a numeric value (for regression).

2. **Choosing "K":** You need to specify the number of nearest neighbors, denoted as "*K*", that the algorithm should consider when making predictions. Selecting an appropriate value for "K" is important, as it can greatly impact the algorithm's performance. A smaller "K" value makes predictions more sensitive to noise, while a larger "K" value makes them smoother.

3. **Calculating Distance:** To find the "K" nearest neighbors of a test data point, the algorithm calculates the distance (e.g., Euclidean distance or Manhattan distance) between that point and every other point in the dataset. The most commonly used distance metric is Euclidean distance, which is computed as the square root of the sum of squared differences between feature values.

4. **Selecting Neighbors:** Once distances are calculated, the algorithm selects the "K" data points with the shortest distances to the test data point, these are the "nearest neighbors."

5. **Majority Voting (Classification):** For classification tasks, KNN counts the number of neighbors belonging to each class among the nearest neighbors and assigns the class label that appears most frequently as the predicted class for the test data point. This process is known as majority voting.

6. **Mean (Regression):** For regression tasks, KNN calculates the mean (average) of the target values of the nearest neighbors and assigns this mean value as the prediction for the test data point.

7. **Prediction:** The algorithm assigns the predicted class label (in classification) or numeric value (in regression) to the test data point based on the majority voting or mean calculation.

8. **Repeat for Each Test Data Point:** Steps 3-7 are repeated for each test data point in the dataset, generating predictions for all test samples.

Despite these limitations, KNN can be a valuable tool in many machine learning applications, especially when we have a relatively small dataset and the data is well-behaved.

3.1 Advantages of KNN

- KNN is easy to understand and implement.
- Since KNN doesn't build an explicit model during training, the training phase is very fast.
- It can be used for both classification and regression tasks.
- KNN doesn't make assumptions about the data distribution.

3.2 Disadvantages of KNN

- The algorithm can be slow, especially when dealing with large datasets because it needs to compute distances for each data point.
- KNN is sensitive to the scale of features, so feature scaling (*normalization/standardization*) is often necessary.
- Choosing the right value for K and the appropriate distance metric can be challenging.
- Storing the entire dataset in memory can be impractical for large datasets.

4 Methods and Materials

4.1 Data Set

The Titanic dataset is a well-known and often-used dataset in the field of data science and machine learning. It contains information about the passengers who were aboard the RMS Titanic during its ill-fated maiden voyage in April 1912. The dataset is frequently used for training and testing machine learning models, particularly for binary classification tasks.

Our dataset consists of 1308 instances and 12 features with some missing values. Here are the key features and details of the dataset:

- **PassengerId:** A unique identifier for each passenger.
- **Survived:** A binary variable indicating whether a passenger survived or not. It has two possible values: 1 for survived and 0 for not survived (died).
- **Pclass (Passenger Class):** The class of the passenger's ticket, which represents their socio-economic status. It can take three values: 1st class (1), 2nd class (2), or 3rd class (3).
- **Name:** The passenger's name.
- **Sex:** The gender of the passenger (male or female).
- **Age:** The age of the passenger.
- **SibSp (Siblings/Spouses Aboard):** The number of siblings or spouses the passenger had aboard the Titanic.
- **Parch (Parents/Children Aboard):** The number of parents or children the passenger had aboard the Titanic.
- **Ticket:** The ticket number.
- **Fare:** The fare paid by the passenger for the ticket.
- **Cabin:** The cabin number where the passenger stayed.
- **Embarked:** The port at which the passenger boarded the ship. It can take three values: C (Cherbourg), Q (Queenstown), or S (Southampton).

4.2 Resources

Equipment:

- Intel i7-9750H @ 2.60 GHz processor.
- 16 GB DDR4 @ 2666 MHz.

- 1 Tb HDD + 1 Tb SSD.
- GeForce GTX 1660Ti 6 GB VRAM GDDR6.

4.3 Libraries

The following libraries were utilized for this activity:

- **NumPy** is a fundamental package for scientific computing in Python as it provides support for large, multi-dimensional arrays and matrices, along with a variety of mathematical functions to operate on these arrays. NumPy is a core library in the data science ecosystem and is used for tasks like numerical computations, linear algebra operations, random number generation, and more.
- **Pandas** is a powerful library for data manipulation and analysis in Python. It provides data structures (primarily Series and DataFrame) to efficiently handle and manipulate structured data. Pandas enables tasks such as loading data from various file formats (CSV, Excel, SQL databases), cleaning and preprocessing data, filtering, merging, transforming, and summarizing data. It's widely used for data wrangling and exploration.
- **Seaborn** is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations like scatter plots, bar plots, histograms, box plots, and more. It also supports color palettes, themes, and additional statistical plotting capabilities.
- **Matplotlib** is a versatile 2D plotting library in Python. It provides a wide range of functions to create static, interactive, and animated visualizations. While it can be used to create basic plots, it can also be customized to create complex visualizations. Matplotlib is the foundation for many other visualization libraries and tools.
- **Scikit-learn** is a machine learning library. It's *preprocessing* module offers various functions for data preprocessing tasks, such as scaling, data imputation, encoding categorical values and much more.

These libraries are essential for various data analysis, data preprocessing, and data visualization tasks in Python.

5 Results

The first step is to import the libraries for this project, as seen on Figure 1.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix, accuracy_score, recall_score
from sklearn.metrics import precision_score, f1_score

from sklearn.model_selection import train_test_split

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

from utils import dimensionality_reduction

pd.options.display.float_format = '{:.2f}'.format

pie_colors = ['#F08080', '#CCCCFF', '#9FE2BF']
countplot_colors = ['#a6b1f7', '#ffcc99', '#DAF7A6']
violinplot_colors = ['#FF5733', '#dbff33', '#ffbd33', '#33dbff']
```

Figure 1: *"Elbow" method.*

Next, we import the dataset to be used and display our data, as shown in Figure 2.

```
Total instances: 1309
Total features: 12
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.25	S	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.925	S	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.1	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.05	S	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
1304	1305	0	3	Spector, Mr. Woolf	male	0.00	0	A.5. 3236		8.05	S	NaN	NaN
1305	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.00	0	0	PC 17758	108.9	C105	C
1306	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.50	0	0	SOTON/O.Q. 3101262	7.25	S	NaN
1307	1308	0	3	Ware, Mr. Frederick	male	0.00	0	359309	8.05	S	NaN	NaN
1308	1309	0	3	Peter, Master. Michael J	male	1.00	1	2668	22.3583	C	NaN	NaN

1309 rows × 12 columns

Figure 2: *Values from the dataset.*

We need to obtain the description of our dataset, shown in Figure 3.

	PassengerId	Survived	Pclass	Age	SibSp
count	1309.00	1309.00	1309.00	1309.00	1309.00
mean	655.00	0.38	2.29	23.97	0.45
std	378.02	0.48	0.84	17.47	0.93
min	1.00	0.00	1.00	0.00	0.00
25%	328.00	0.00	2.00	8.00	0.00
50%	655.00	0.00	3.00	24.00	0.00
75%	982.00	1.00	3.00	35.00	1.00
max	1309.00	1.00	3.00	80.00	9.00

Figure 3: *Values from the dataset.*

The next step involves checking our dataset for any missing values, Figure 4.

```
PassengerId       0
Survived          0
Pclass            0
Name              0
Sex               0
Age               0
SibSp             0
Parch             0
Ticket            0
Fare              0
Cabin           241
Embarked       1039
dtype: int64
```

Figure 4: *Missing values.*

Next, we obtain the type of data on our dataset, Figure 5.

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   int64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   object
 8   Ticket       1309 non-null   object
 9   Fare         1309 non-null   object
 10  Cabin        1068 non-null   object
 11  Embarked     270 non-null    object
dtypes: float64(1), int64(4), object(7)
memory usage: 122.8+ KB
```

Figure 5: *Types of data.*

After further inspecting some of our features, we notice that the ”*Fare*” and ”*Parch*” features have some unnecessary characters, so in order to remove them, we create a dictionary, fill the empty data with *NaN*, convert this data into a float data type, lastly, we fill this *NaN* data with it’s median value, results shown in Figure 6.

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   int64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   float64
 8   Ticket       1309 non-null   object
 9   Fare         1309 non-null   float64
 10  Cabin        1068 non-null   object
 11  Embarked     270 non-null    object
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

Figure 6: *Types of data after data cleaning.*

Figure 7 shows a heatmap of our missing data.

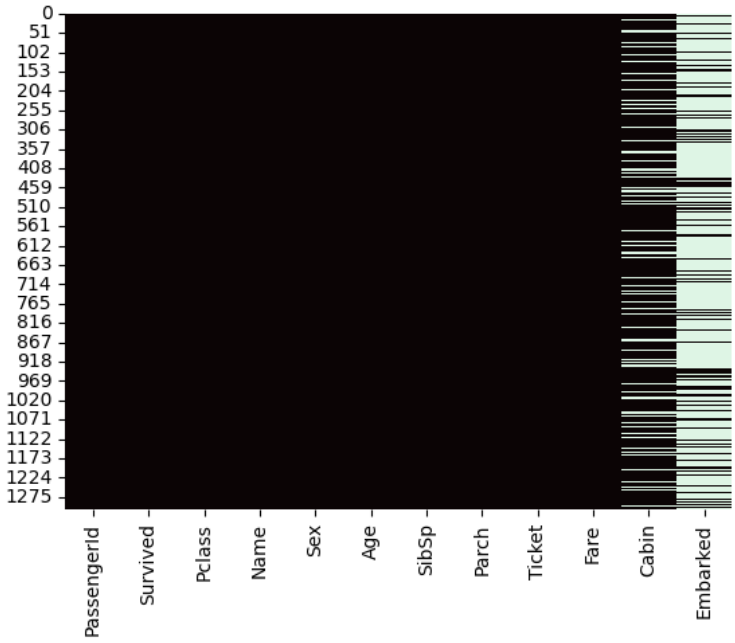

Figure 7: *Types of data after data cleaning.*

For the distribution of our data, we will use a pie chart in order to acknowledge the survival probability across our database's features, starting with the "*Survived*" percentage, shown in Figure 8.

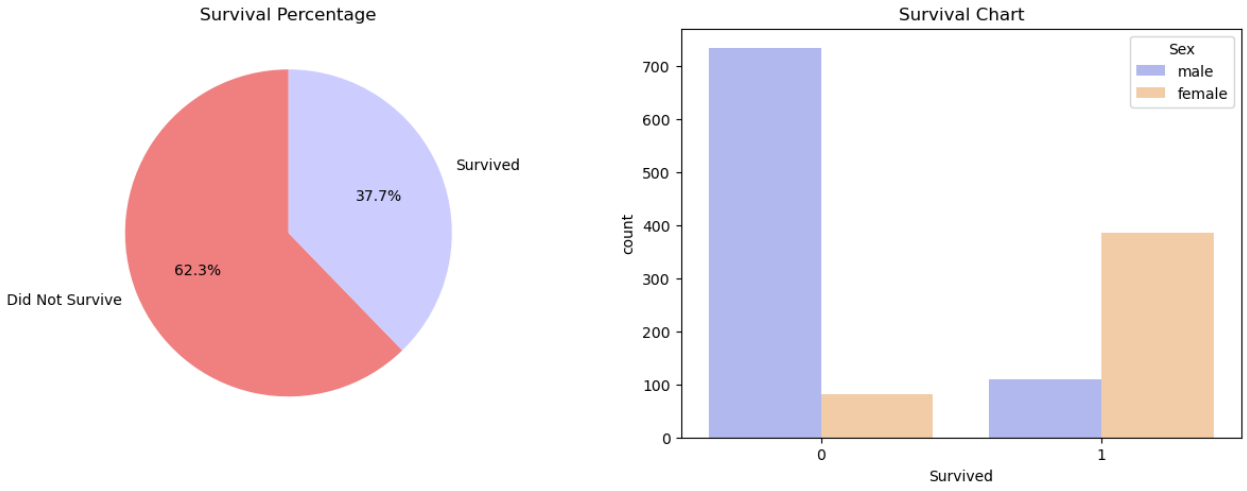

Figure 8: *Survival Percentage.*

Next, the gender survival probability, Figure 9.

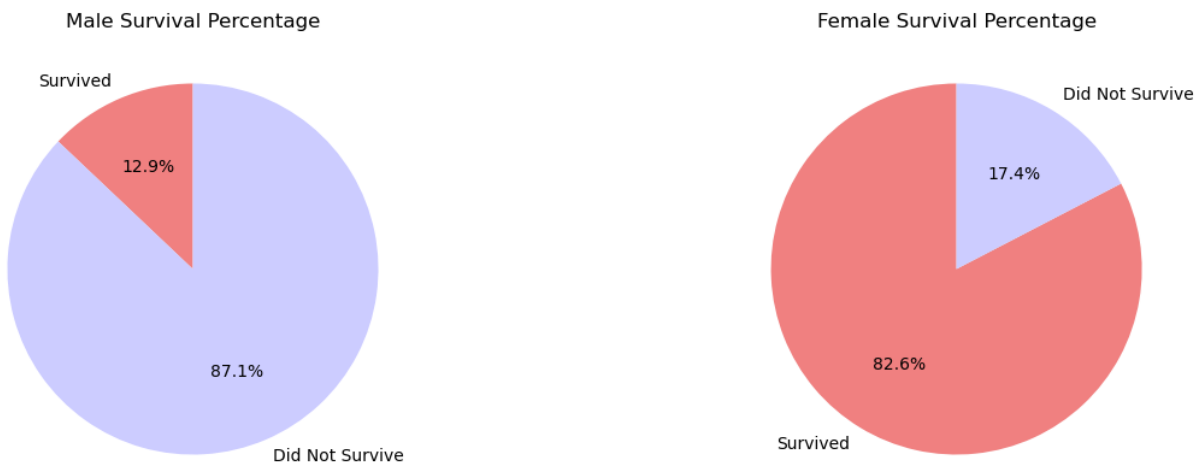

Figure 9: *Gender Survival Percentage.*

Next, the Passenger Class distribution, Figure 10.

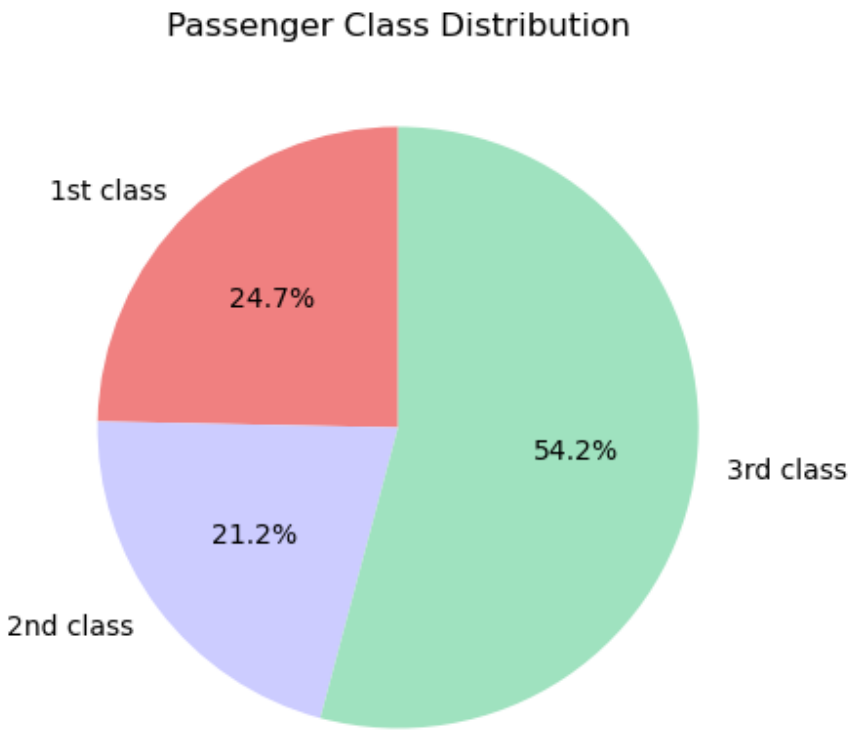

Figure 10: *Passenger Class Survival Distribution.*

Figures 11 & 12 show the Passenger Class survival.

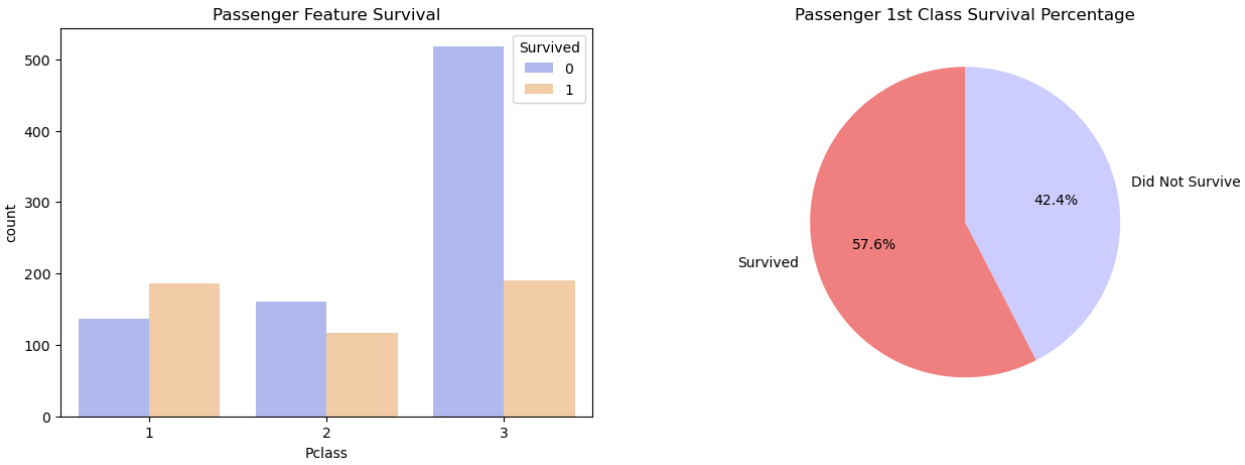

Figure 11: *Passenger Class Survival Percentage.*

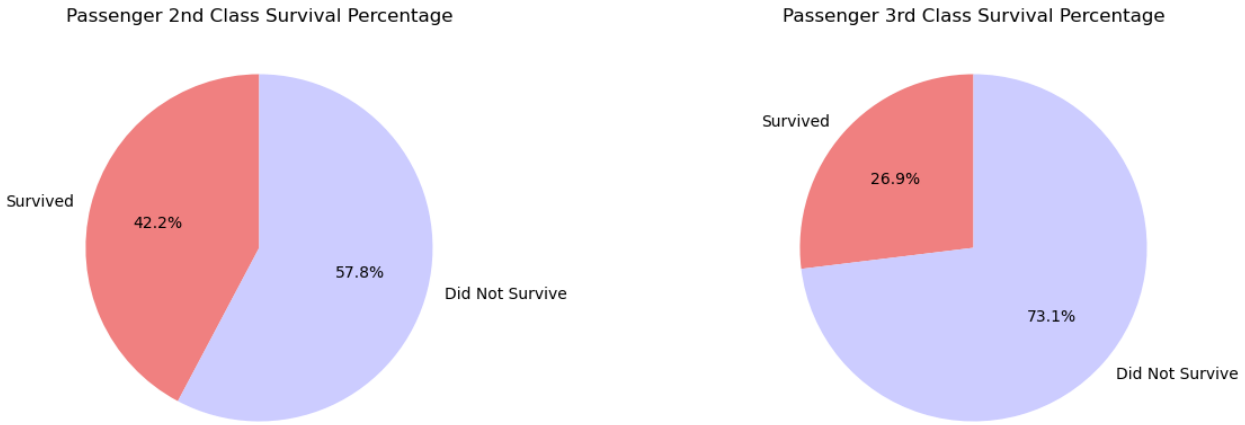

Figure 12: *Passenger Class Survival Percentage.*

Figures 13 & 14 show the survival percentage w.r.t. the "*Embarked*" feature.

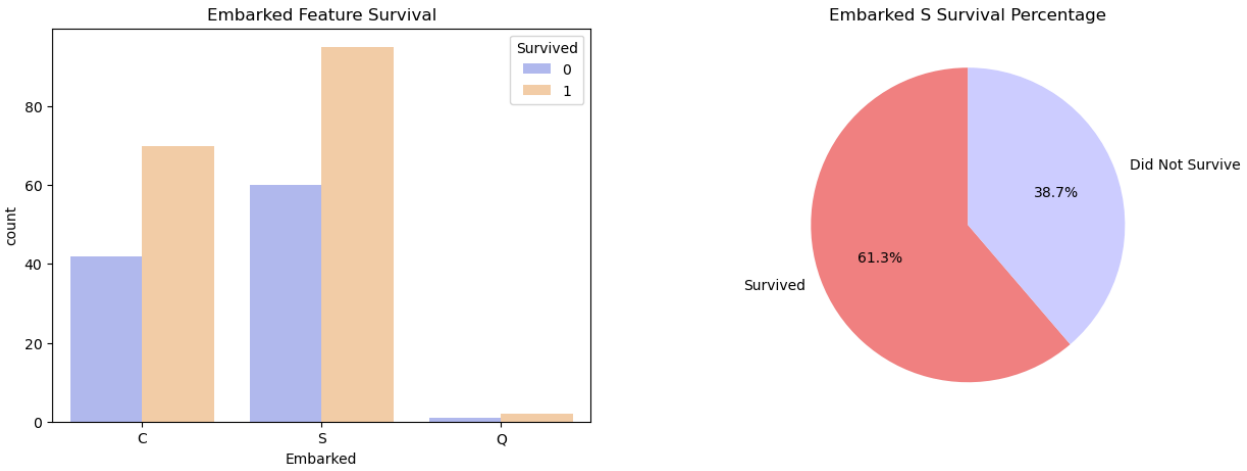

Figure 13: *Embarked Survival Percentage.*

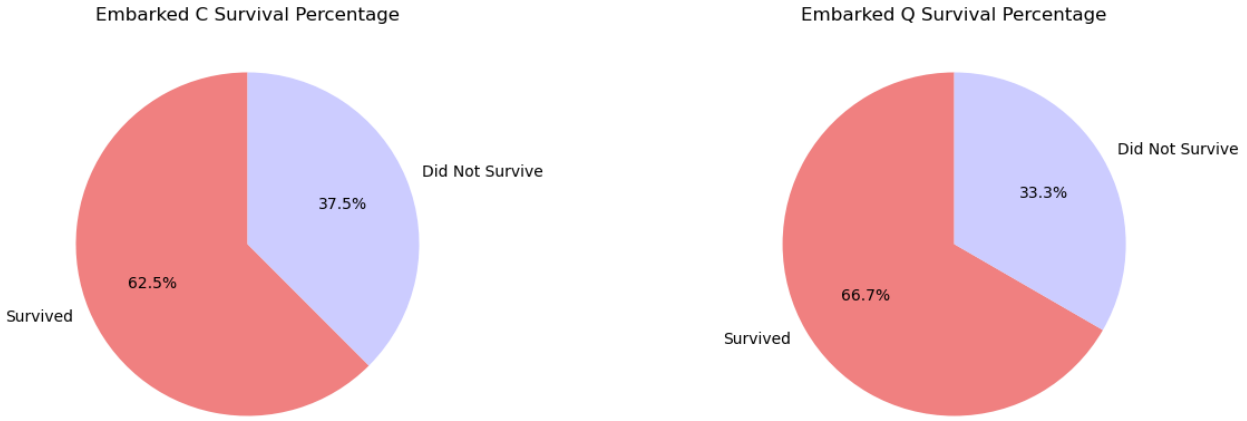

Figure 14: *Embarked Survival Percentage.*

Figure 15 shows the distribution of the ”*Fare*” feature.

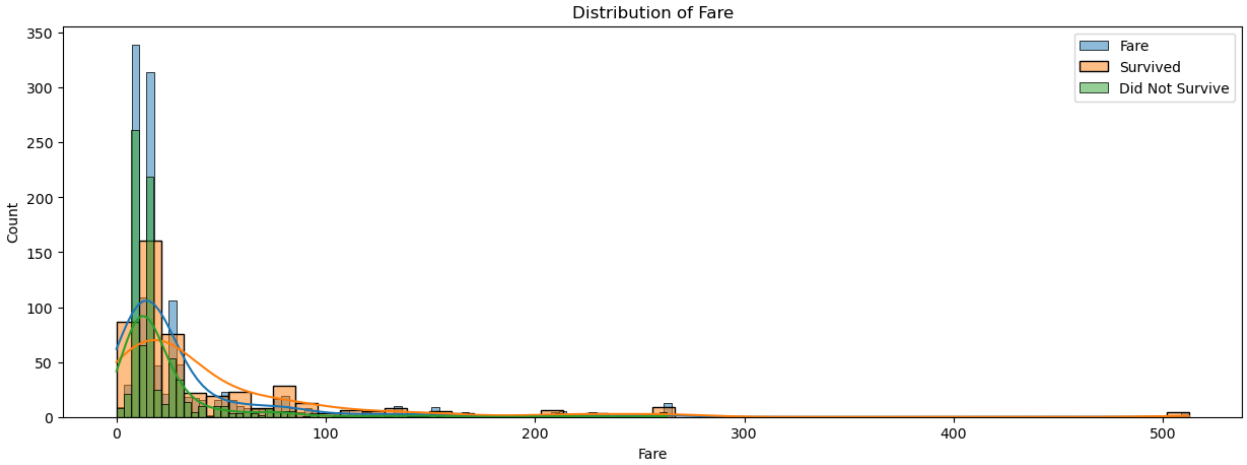

Figure 15: *"Fare" Distribution.*

For the survival percentage on the ”*SibSp*” & ”*Parch*” features, Figure 16 shows a bar graph.

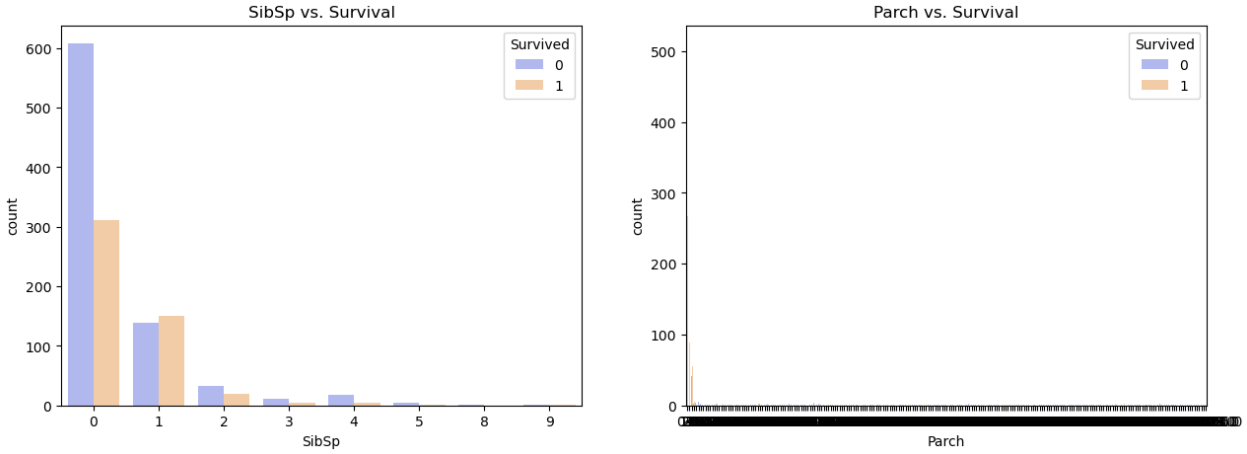

Figure 16: *"SibSp" & "Parch" Survival Percentage.*

Next, we need to know the survival distribution for certain age groups, shown on Figure 17.

- **Group 0:** ages from 0 to 21.
- **Group 1:** ages from 22 to 41.
- **Group 2:** ages from 42 to 61.
- **Group 3:** ages from 62 to 81.

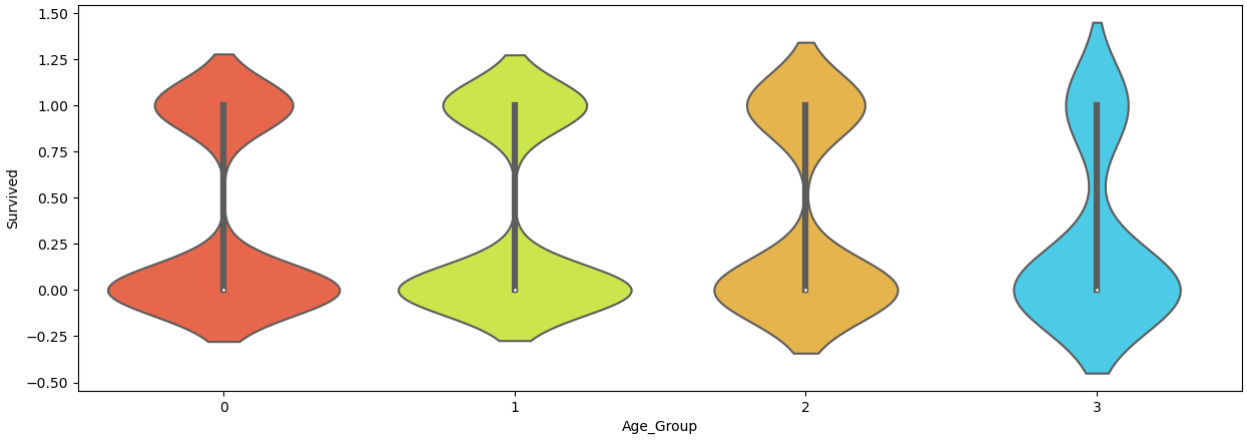

Figure 17: *Age groups survival distribution.*

Figure 18 shows the ”*Fare*” and ”*Age*” distribution w.r.t. ”*Survived*” feature.

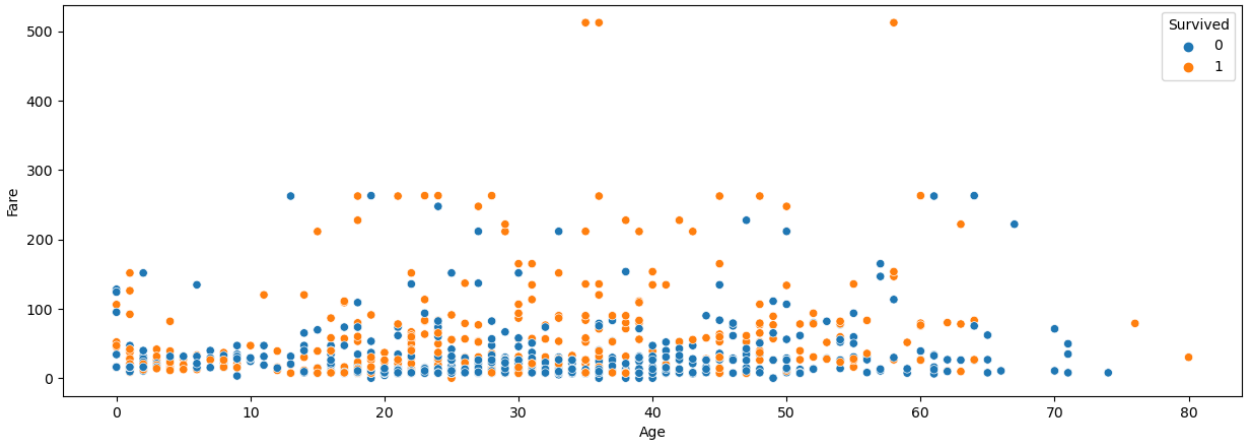

Figure 18: *"Fare" and "Age" survival distribution.*

Figure 19 shows the distribution of the ”*Title*” features w.r.t. the *Survived*” feature.

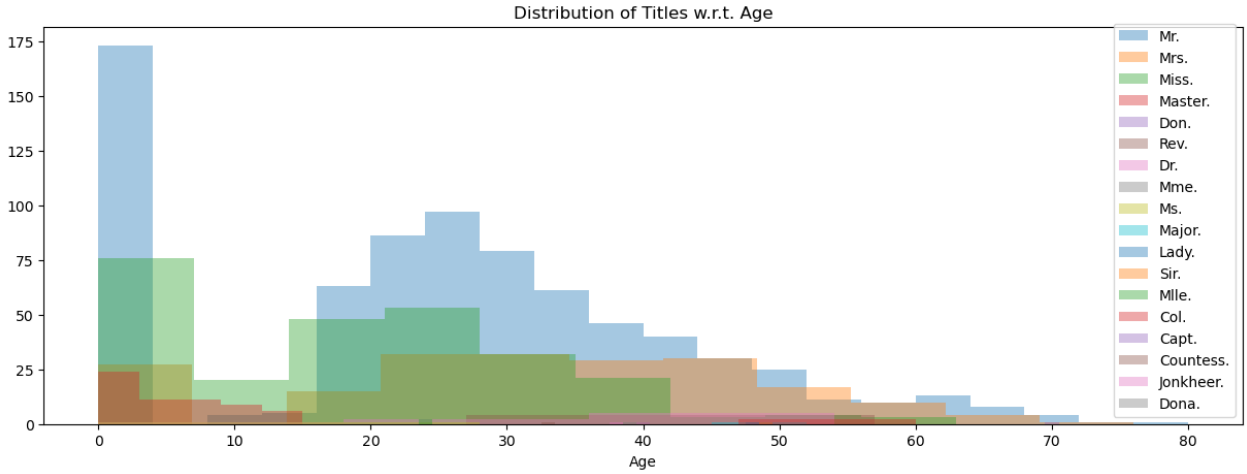

Figure 19: *Distribution of Titles w.r.t. Age.*

Figure 20 shows the distribution of the ”*Age Group*” features w.r.t. the ”*Survived*” feature.

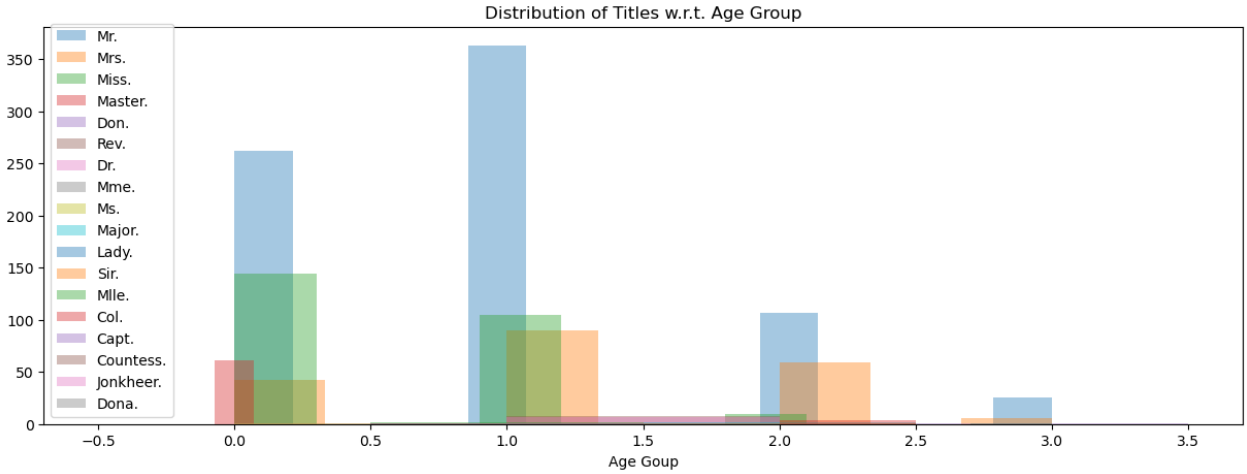

Figure 20: *Distribution of Titles w.r.t. Age.*

Figure 21 shows the distribution of the "*Titles*" features w.r.t. "*Survived*" feature.

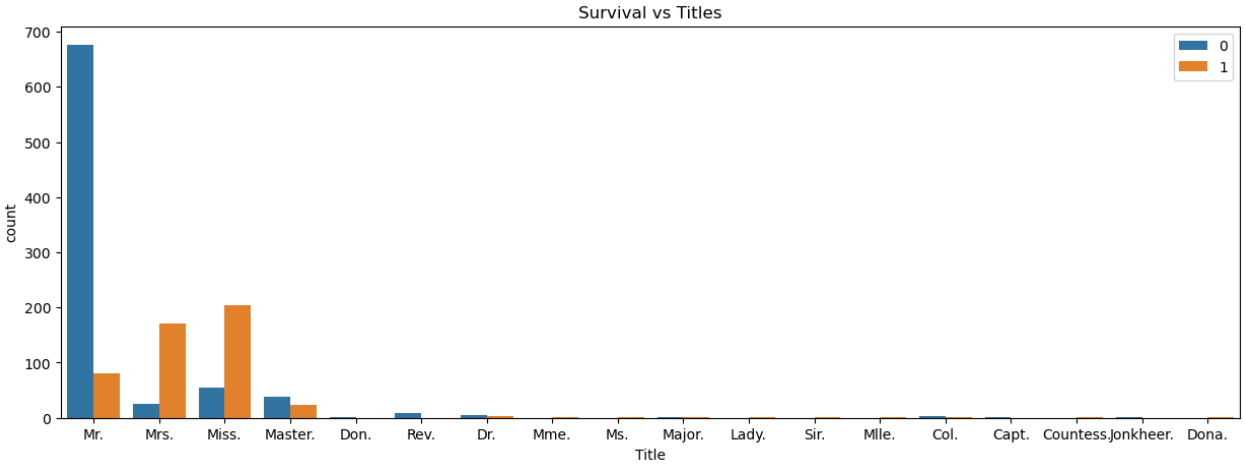

Figure 21: `Distribution of Titles w.r.t. Survival.`

Insights from the EDA

- Female passengers were saved more than male passengers.
- Passenger class was considered while saving the passengers.
- Port of embarkation was not considered.
- "*Parch*" & "*SibSp*" don't give much information.
- Distribution of "*Fare*" shows preference was given to passengers who paid more for their tickets, probably because of class.
- "*PassengerId*","*Ticket*" and "*Cabin*" features will be dropped since they are not necessary.

For the next part, we need to make a Feature Selection, in order to accomplish this, we need to convert our categorical data (*"Name", "Sex"*) into numerical values, also adding any missing values from the "*Embarked*" feature filling them with "*Q*".

Next, we obtain a Pearson correlation between different features from our dataset, seen in Figure 22.

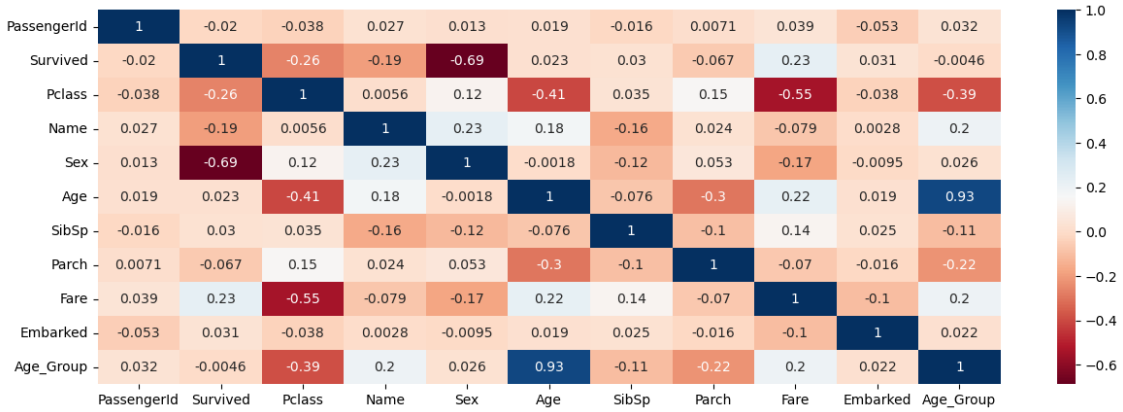

Figure 22: *Heatmap.*

Now we proceed to delete unnecessary features from our data set, the result is shown in Figure 23.

	Survived	Pclass	Sex	Age	Fare
0	0	3	1	22.00	7.25
1	1	1	0	38.00	71.28
2	1	3	0	26.00	7.92
3	1	1	0	35.00	53.10
4	0	3	1	35.00	8.05

Figure 23: *Dropped data.*

The next step, is to utilize the Chi-squared test to assess how well the frequency distribution fits an expected distribution. It will help us to determine if observed data follows a specified distribution, Figure 24.

	Column	Score
3	Fare	5424.52
1	Sex	220.82
0	Pclass	28.04
2	Age	8.46

Figure 24: *Chi-squared test.*

Insights from the Feature Selection

- Features like ”*Pclass*”, ”*Name*”, ”*Sex*” and ”*Embarked*” show a negative correlations with relation to ”*Survived*”.
- ”*Fare*” feature shows positive correlation with relation to ”*Survived*”.
- No other feature shows correlation to ”*Survived*”.
- ”*Fare*” and ”*Pclass*” show negative correlation with each other because of the class system.
- For prediction purposes, ”*Fare*”, ”*Pclass*”, ”*Sex*” and ”*Age*” features will be used.

For this analysis, we will use a manual selection based on the results obtained from our chi-squared test and the Pearson correlation to select the best features for our prediction.

In the process of manual feature selection, we opted for the ”*Sex*”, ”*Age*”, and ”*Pclass*” features. For the Pearson feature selection, we identified the three features with the highest correlation to the ”*Survived*” feature, which were ”*Sex*”, ”*Fare*”, and ”*Pclass*”.

Figure 25 shows part of the code created for the *KNN* algorithm.

```python
class KNNClassifier:

    def __init__(self, k=3, distance_metric="euclidean"):
        self.k = k
        self.distance_metric = distance_metric
        self.X_train = None
        self.Y_train = None

    def fit(self, X_train, Y_train):
        self.X_train = X_train
        self.Y_train = Y_train

    def predict(self, X_test):
        predictions = []
        for x_test in X_test:
            if self.distance_metric == "euclidean":
                distances = np.linalg.norm(self.X_train - x_test, axis=1)
            elif self.distance_metric == "manhattan":
                distances = np.sum(np.abs(self.X_train - x_test), axis=1)

            sorted_indexes = distances.argsort()[:self.k]
            sorted_labels = self.Y_train[sorted_indexes]
            predicted_label = np.bincount(sorted_labels).argmax()
            predictions.append(predicted_label)
        return np.array(predictions)
```

Figure 25: *Code for KNN algorithm.*

Figures 26 and 27 show the results after evaluating our model with the selected features from the Pearson Correlation heatmap.

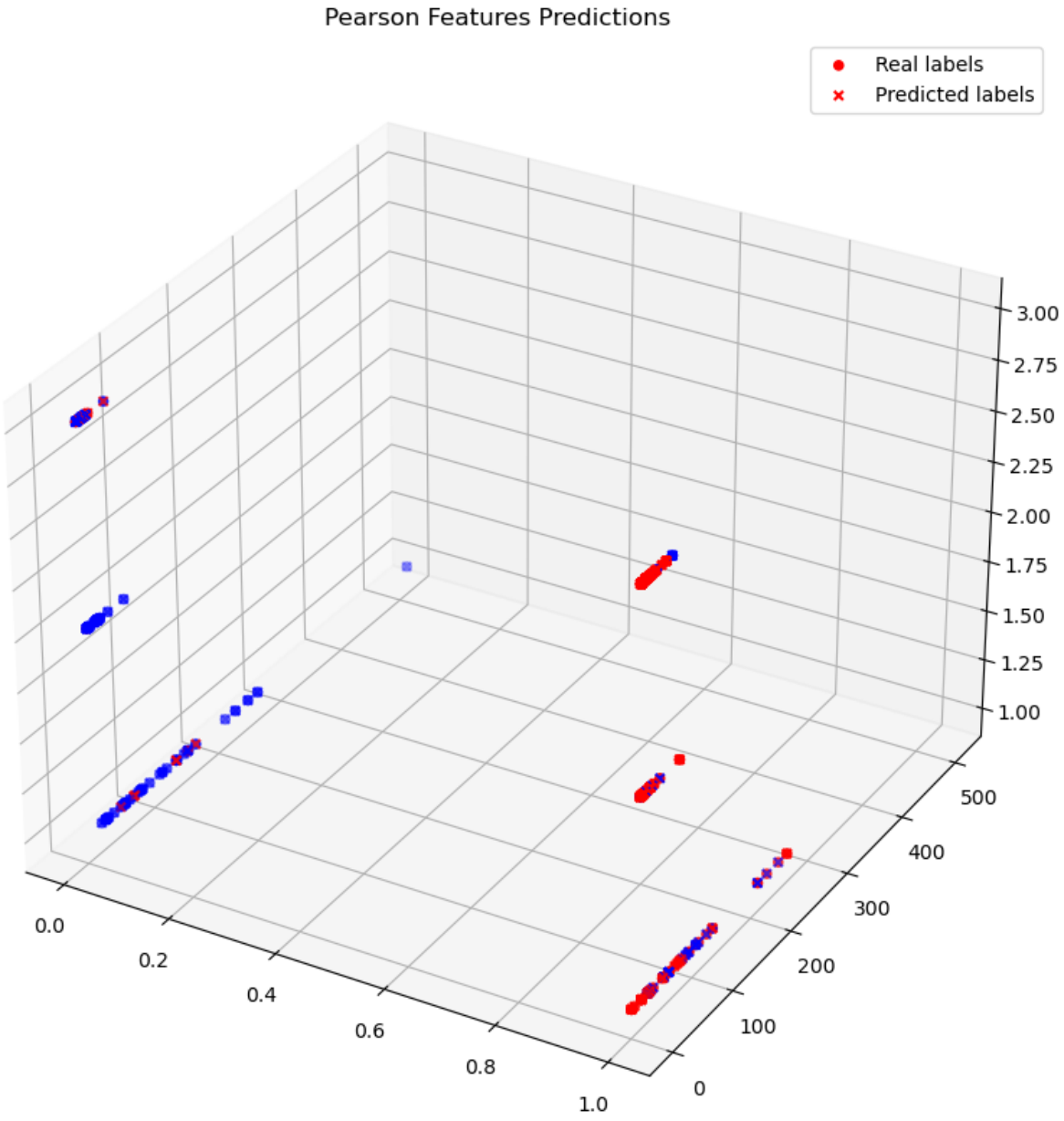

Figure 26: *Pearson Prediction.*

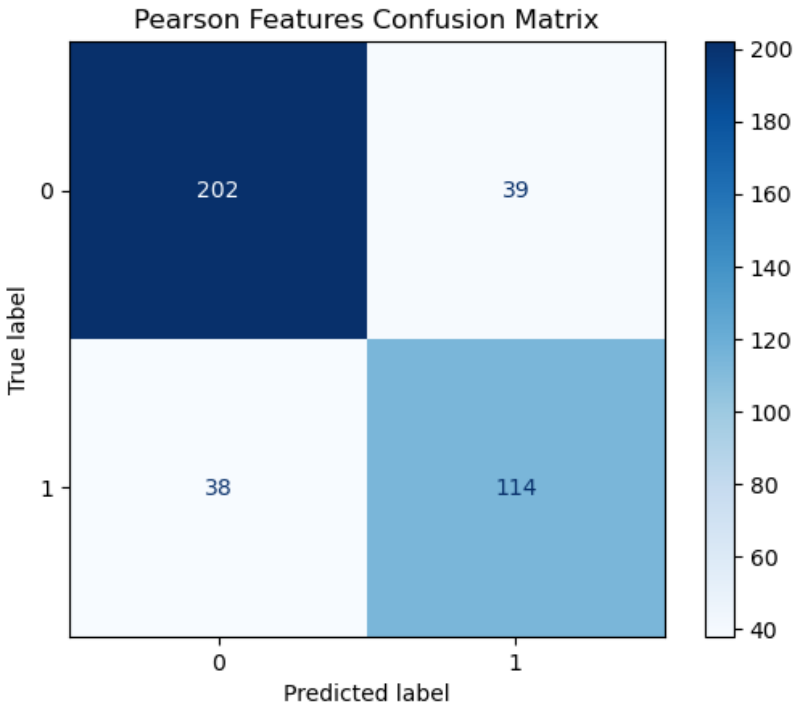

Figure 27: *Pearson Confusion Matrix.*

Pearson Features Results	
Accuracy	80.41%
Recall	75.00%
Precision	74.51%
F1 Score	74.75%

Table 1: *Results from Pearson feature selection.*

Table 1 shows the results from the evaluation with features taken from the Pearson Correlation heatmap.

Figures 28 and 29 show the results after evaluating our model with the selected features from the Pearson Correlation heatmap.

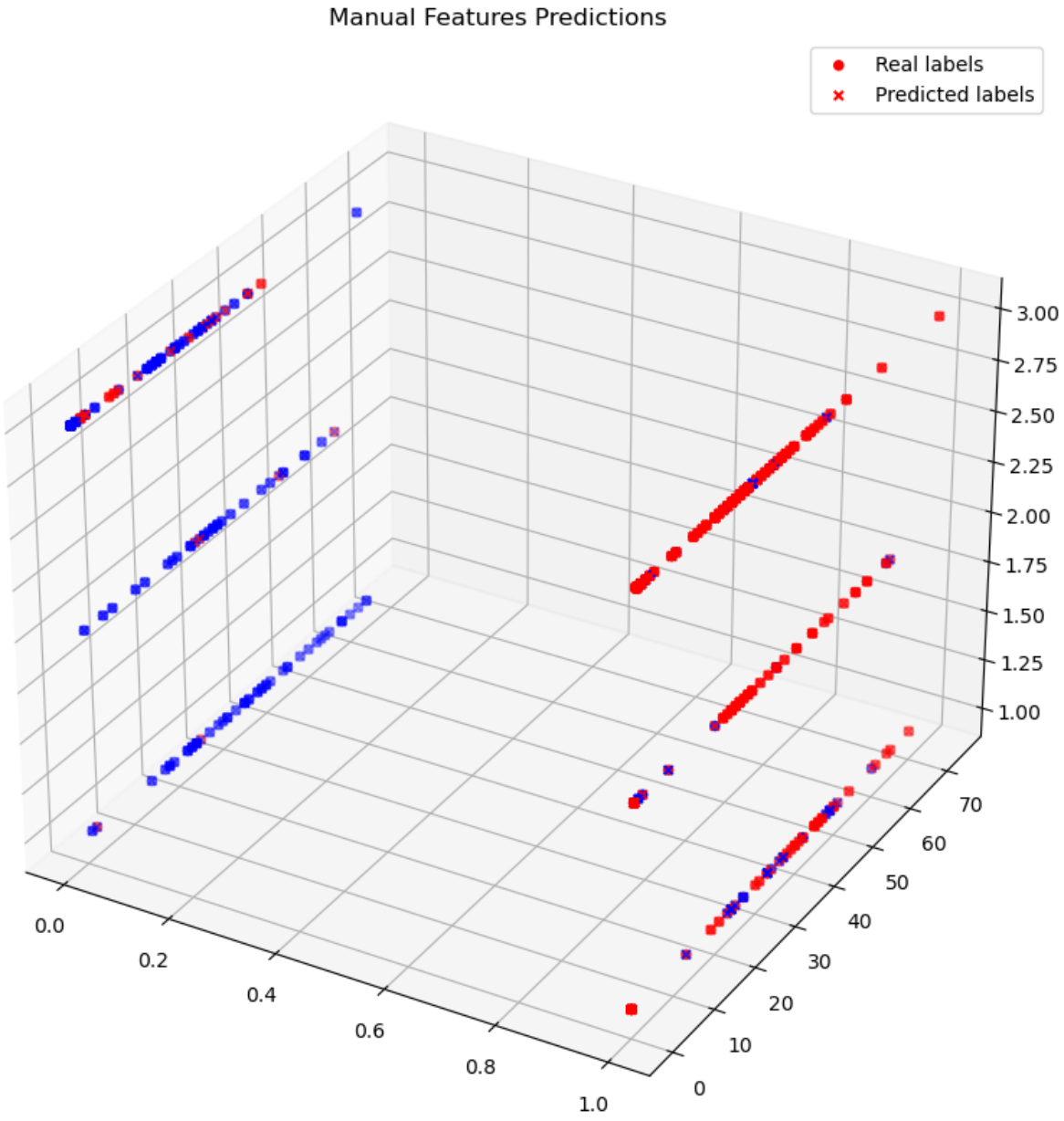

Figure 28: *Manual features selection Prediction.*

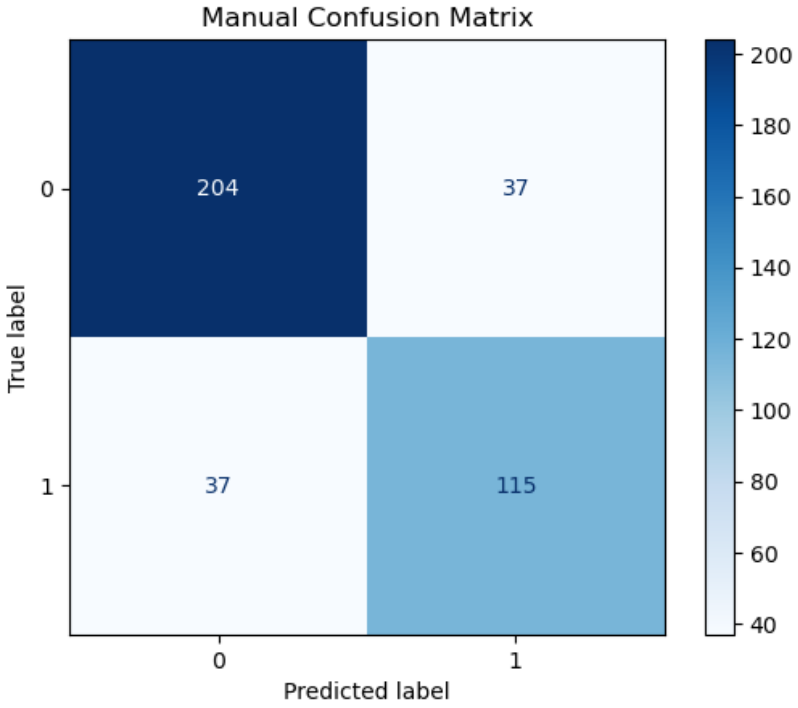

Figure 29: *Manual features selection Confusion Matrix.*

Manual Features Results	
Accuracy	81.17%
Recall	75.66%
Precision	74.66%
F1 Score	74.66%

Table 2: *Results from Manual features selection.*

Table 2 shows the results from the evaluation with features manually selected.

6 Discussion

KNN is a simple and easy-to-understand algorithm; it's a non-parametric method that doesn't make strong assumptions about the underlying data distribution, which makes it a good choice, especially for introductory machine learning projects, and since our dataset involves binary classification, this algorithm is well-suited for this particular task.

KNN focuses on local patterns in the data, for example, it identifies passengers with similar characteristics (e.g., age, class, and gender) and predicts survival based on the majority class within that local group. This can be useful if survival is influenced by specific clusters of passenger characteristics.

For this activity, we noticed that manually selecting the features, yielded better results than those obtained from the Pearson Correlation heatmap.

These results suggest that the attribute selection technique based on Pearson's correlation coefficient may also be a valuable tool for survival detection.

The high accuracy, sensitivity, and precision achieved by these techniques indicate that it has the potential to effectively identify Titanic accident survival and minimize false positives and false negatives. However, further analysis and validation with additional data sets is needed to confirm the utility and generality of these results.

It's important to note that while KNN has its advantages, it also has limitations, such as sensitivity to the choice of the number of neighbors and the distance metric.

7 Conclusions

Our exploration of the Titanic dataset and the application of the KNN algorithm represent a fundamental step in data analysis and machine learning. While KNN served as a strong baseline model for binary classification, it is essential to acknowledge its limitations, including sensitivity to the choice of k and distance metric. To further enhance prediction accuracy, advanced machine learning techniques and feature engineering could be explored.

KNN can serve as a baseline model for comparison with more advanced machine learning algorithms since it provides a starting point for modeling and helps establish a benchmark for performance evaluation.

In conclusion, this activity provided a compelling glimpse into the Titanic tragedy, offering a data-driven perspective on the passengers' stories. It is a testament to the power of data science in uncovering historical insights and paying homage to those who were aboard the ill-fated voyage.

References

[1] L. Pierson, *Data Science for Dummies*. Wiley, 2015.

[2] M. Aceves, *Inteligencia Artificial Para Programadores Con Prisa*. Universo de Letras, 2021.

[3] D. Little, R. Rubin, *Statistical Analysis with Missing Data*, 3rd ed. Wiley, 2019.

[4] "BD Titanic completa," 2023, Provided by Aceves, M.

[5] W. Shahzad, Q. Rehman, and E. Ahmed, "Missing Data Imputation using Genetic Algorithm for Supervised Learning," *International Journal of Advanced Computer Science and Applications*, vol. 8, 2017.

[6] scikit-learn developers, "6.4 imputation of missing values," 2023, Scikit Learn. [Online]. Available: https://scikit-learn.org/stable/modules/impute.html

[7] IT 240 Course, "Normalization: Relational database normalization process," 2023, DePaul University. [Online]. Available: https://condor.depaul.edu/gandrus/240IT/accesspages/normalization3.htm