Titanic: Machine Learning from Disaster

*Kalee Crandall, Beth Maurer, Chris Burgers*

[Kalee.Crandall@trojans.dsu.edu, Beth.Maurer@trojans.dsu.edu, Chris.Burgers@trojans.dsu.edu](mailto:Kalee.Crandall@trojans.dsu.edu,%20Beth.Maurer@trojans.dsu.edu,%20Chris.Burgers@trojans.dsu.edu)

**1. Task**

The Titanic was considered an unsinkable ship. Unfortunately, it collided with an iceberg on its maiden voyage and sank on April 15, 1912. This infamous shipwreck resulted in the death of 1,502 passengers and crew.

Our task (see Figure 1) is to determine which groups of people aboard the ship were more likely to survive than others. We will decipher if survival was purely a result of luck or if other factors were at play. To do this, we will use machine learning to create models that can predict which passengers survived this tragic event.

Some factors that might make this challenging are missing values within the data. The data for the cabin in which the passengers were assigned, some of their ages, and their fare are missing values, which will need to be cleaned up. Additionally, making the decision of what columns are valuable and invaluable can be challenging.



Figure 1. Our task is to predict survival on the Titanic.

**2. Approach**

To accomplish this challenging task, we will be using a data set comprised of Titanic passenger information to predict survival for each passenger. This data set will be fed through various machine learning models that are implemented using the Python programming language.

The Python programming language, Python libraries, and machine learning models will be displayed through Jupyter notebooks. Jupyter notebooks are utilized as an interactive environment for Python that can show all the steps taken through code and images. This will make the process easy for the user to follow in case of questions or concerns about the results.

We will be using the following Python libraries: Sklearn (for machine learning algorithms and preprocessing), Numpy, Pandas, and Matplotlib (for graphing). These libraries will be used to fit, predict, and score the accuracy of the machine learning models. The following models will be utilized: k-Nearest Neighbors, Decision Trees, Logistic Regression Classification, Artificial Neural Networks (ANNs), SVMs, and XGBoost.

Finally, we will use cross-validation and hyperparameter tunning to determine which model most accurately predicts the survivors on the Titanic.

Below are a few examples of the equations that are used to produce the predictions of a few of the models we implemented.

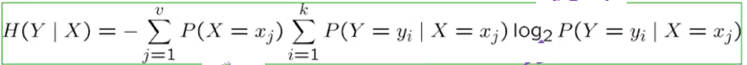
**kNN Diagrams & Equations**

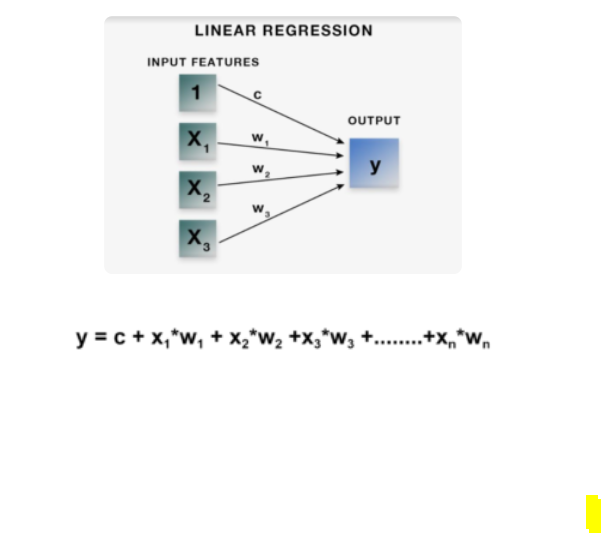
Euclidean Distance: Table

Description automatically generated

**Decision Tree Diagram & Equations**

Information Gain:





**3. Dataset**

Our data set is the Titanic data set from [Kaggle](https://www.kaggle.com/c/titanic/data). It separates the passengers into a training set of 891 people and a testing set comprised of 418 people.

The data set contains 12 features for each passenger. Specific information includes the passenger’s class of ticket (1st, 2nd, or 3rd), their sex (male or female), the age of the passengers, their family size (siblings, spouses, parents, and children), a few ticket details like ticket number and cost, cabin number, and finally the port of embarkment.

We will perform data preprocessing to convert some of the categorical metrics (like sex, ticket class, and port of embarquement) into numerical features. This will be accomplished using Sklearn’s OneHotEncoder functions. The passenger data will also need to be cleaned up to remove missing or null values that are found in columns like Embarkment. In the case of missing values for Age and Fare, we will use the median value of these fields. The reason for using the median rather than the mean is the median is less likely to be influenced by outliers within the dataset.

Our primary measure of success for each machine learning model will be the test set classification accuracy, defined as the number of correct predictions made as a ratio of all predictions made.

**4. Results**

To begin this project, we loaded the data into a Pandas data frame and did some preliminary data exploration. From this, we learned that data preprocessing will be needed for the Age and Embarked features. This is because only 714 of the 891 test cases contain age information and 2 test cases are missing Embarked information. A third column, Cabin information, had missing data for all but 204 of the passengers. Since this data does not appear to be predictive of survival, we excluded this feature altogether.

Additional data discovery led to the understanding that only 342 of the 891 passengers in our test set survived. So, there was a 61.6% mortality rate. See figure 2.

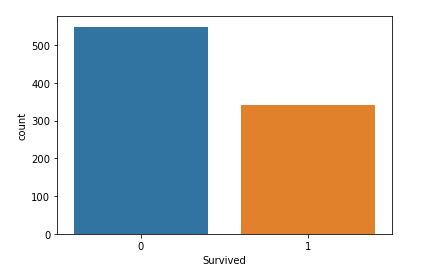


Figure 2: Survival in Orange (1) and death in Blue (0).

Of note though is how mortality rates differed greatly based on sex. By looking at passenger data for females and comparing the number of females on board to the number of females that survived, we determined that 74.2% of women survived. In contrast, when the same procedure was performed for men, we learned that only 18.9% of males survived. See figure 3.

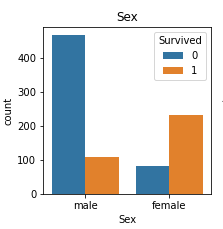


Figure 3: Death in blue (0) and survival in orange (1).

The final preliminary studies we performed were around the financial status of each passenger. We suspected that more affluent passengers would have a higher likelihood of surviving. Sadly, this appears to be the case, as survival rates declined rapidly between 1st, 2nd, and 3rd class passengers. See figure 4.

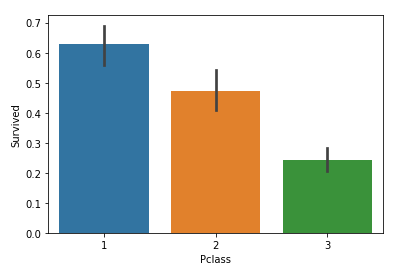


Figure 4: Survival rates of 1st class (blue), 2nd class (orange), and 3rd class (green) passengers.

With these high-level insights in mind, we looked to unlock the predictive power of Sklearn’s machine

learning modules to develop a deeper understanding of what factors impacted passenger survival.

Below we first explored some of the features that can be find within the dataset. This helped us better understand the features and process our results. A picture of the graph along with the code used to explore the features are included along with a brief description of each figure.

*Embarkment Distribution*

Figure 5 shows us that most passengers embarked from Southampton (S). So, we will replace the 2 null values with S. This is important because the null values could skew the data.

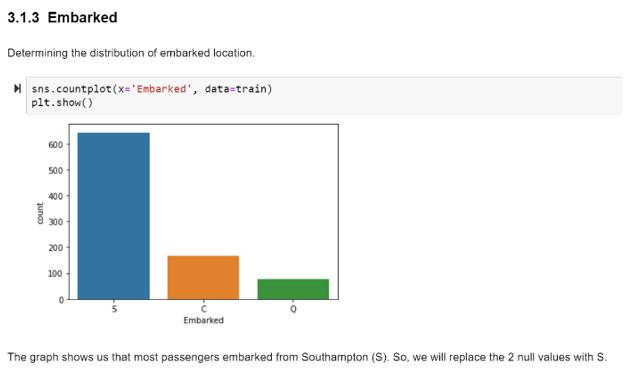


Figure 5: Embarkment location of passengers.

Figure 6 displays the number of survived and dead, coloring based on gender. We noticed in our preliminary results that more women survived than men, which is supported by a deeper dive and displayed in the graph below.

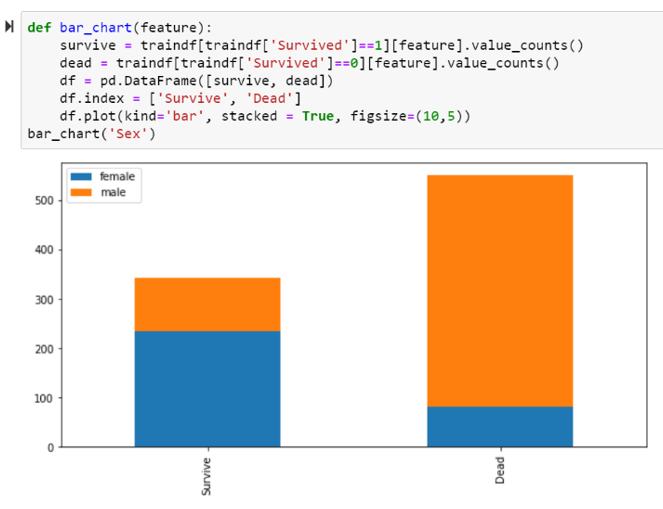


Figure 6: Survival rates based on gender, female (blue), male (orange).

Figure 7 explores the correlation between the features. The correlation of the features is important because it can assist in the understanding of our results and feature selection for each method.

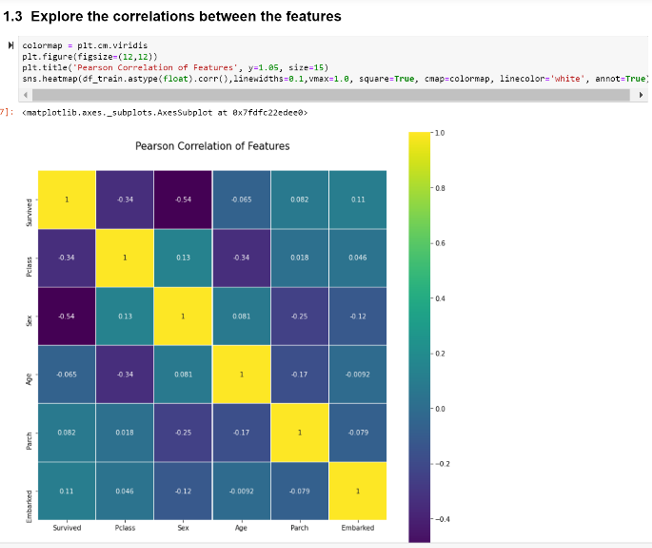


Figure 7: Correlation between the features.

*k-Nearest Neighbors (kNN)*

kNN is considered a lazy learning model and does not handle a large number of features well. Because of this, to implement kNN the most important features needed to be determined. The method SelectKBest was used to calculate the most influential features to use in the model as seen in Figure 8.

Table

Description automatically generated

Figure 8: Best Features determined using SelectKBest

The appropriate number of neighbors was also calculated as shown in Figure 9 and it was determined that 4 neighbors should be used in the model.

Graphical user interface, text, application, email

Description automatically generated

Figure 9: Code showing the appropriate number of neighbors.

The next portion of figures displayed reflect the scores given based on our results that are pushed to a csv for submission to the Kaggle website. Kaggle provides the opportunity to receive a score on our submission csv files which contain the results that came from our predicted machine learning methods.

Using only two features (Sex and Pclass) to implement the kNN model produced an accuracy score of only 76% as shown in Figure 5. We learned that by adding an additional feature, “Embarked” we could improve the accuracy score from 76% to almost 79% as shown in Figure 10.

Graphical user interface, application

Description automatically generated

Figure 10: Accuracy scores of kNN.

*Decision-Tree*

To build the decision-tree, we used features as determined in the kNN model, and then used Cross-Validation to determine the appropriate depth of the decision tree (Figure 11).

Table

Description automatically generated

Figure 11: Determining the appropriate depth of the decision-tree using cross-validation.

Figure 12 shows the final decision-tree with a depth of 3. We can also see that sex is the first feature included on the tree.

Timeline

Description automatically generated

Figure 12: Final decision-tree

*Logistic Regression*

To determine which features should be kept in our Logistic Regression analysis, we used recursive feature elimination with cross validation to reduce the number of features we would use. This helped us determine that the optimal number of features was 8, with "Fare" being excluded. The predictive impacts of Fare appear to be better captured by Class. So we analyzed the following features: 'Age', 'SibSp', 'Parch', 'Pclass\_2', 'Pclass\_3', 'Embarked\_Q', 'Embarked\_S', 'Sex\_male'.

Next, we fine-tuned the model parameters. For this passenger survival task, we chose to fine tune the regularization parameter C. This parameter determines the strength of regularization of the model. Higher values of C correspond to less regularization, or less generalization to out of sample data. On the other hand, low values of C put more emphasis on producing coefficients that are closer to zero. Rather than manually changing values of C to test model results, we used GridSearchCV to do the testing for us.

After feature selection and parameter tuning, the Logistic Regression model produced an testing accuracy score of 0.77511 per Kaggle. See Figure 13.

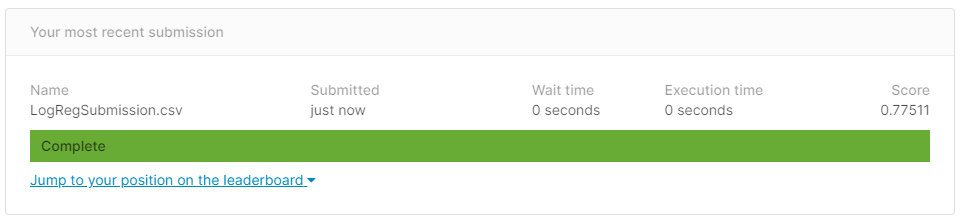


Figure 13: Logistic Regression Score

*Artificial Neural Network*

Next, we tried to improve on our Logistic Regression testing accuracy results by using Artificial Neural Networks (ANNs). We chose to use a multiplayer perceptron (MLP) model to see if it could produce better predictions of survival. After scaling the data to work with the MLP, the MLP model produced an testing accuracy score of 0.77990 per Kaggle. See Figure 14.

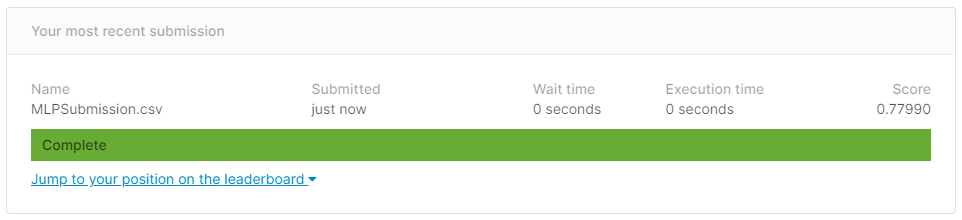


Figure 14: MLP Score

The final portion of our neural network analysis involved trying to fine tune the MLP model to improve the score we received on Kaggle. This approach was similar to how we fine-tuned the regularization parameter C on the Logistic Regression model. Here we tuned the solver, alpha, and hidden\_layer\_size of the MLP model. These parameters change how the model makes predictions, and therefore could potentially improve the strength of the model. This tuning was done using Grid Search.

Surprisingly, the mlpTuned model did not outperform the original MLP model or the Logistic Regression model. Even though it performed better on the training data set, the mlpTuned model did not generalize as well to the testing data set. It ultimately received a score of 0.77272 on Kaggle. See Figure 15.

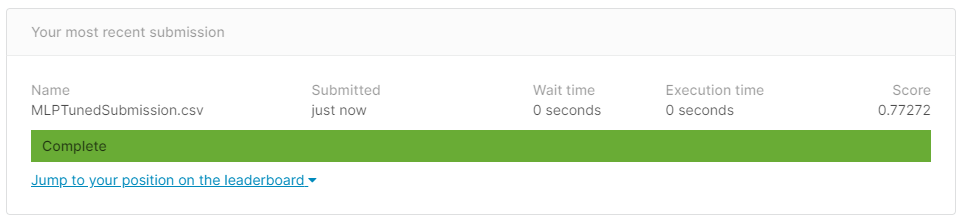


Figure 15: MLP Tuned Score

*XGBoost*

We also explored using XGBoost to improve the survival predictions on the testing data set. The model parameters were fine-tuned using SKLearn’s RandomizedSearchCV to improve our predictions. When the resulting output was uploaded to Kaggle, the accuracy score dropped to 0.76555. See Figure 16.

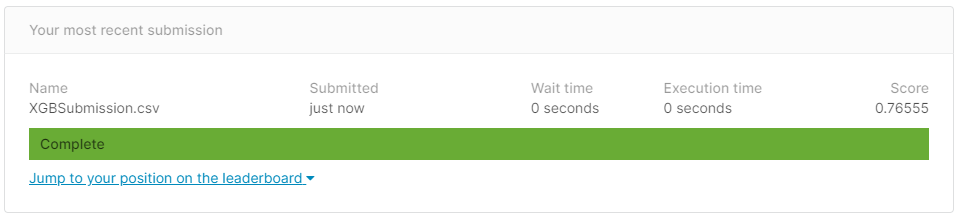


Figure 16: XGBoost Score

*Support Vector Machine (SVM)*

The Support Vector Machine (SVM) classifier was also used to help with classification and regression problems within set. The predicted survival and death numbers were learned through the SVM classifier and analyzed by a performed testing set hosted by Kaggle. This produced an accuracy score of 0.78229. See Figure 17.

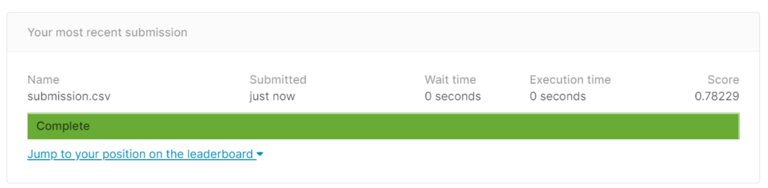


Figure 17: SVM Score

Overall, this is how the machine learning models ranked from a test set accuracy perspective:

1. Decision Tree: 0.80500
2. KNN (3 features): 0.78947
3. Support Vector Machine: 0.78229
4. ANN (via MLP): 0.77990
5. Logistic Regression: 0.77511
6. MLP (tuned): 0.77272
7. XGBoost: 0.76555
8. KNN (2 features): 0.76555

**5. Detailed Timeline and Roles**

|  |  |  |
| --- | --- | --- |
| **Task** | **File Names** | **Lead** |
| Data Preprocessing | Milestone 2 – INFS 768 – Beth – Chris – Kalee.docx | All |
| Implement kNN: including cross-validation and hyperparameter tunning. | kNN Predictions Titanic Dataset.ipynb | Kalee |
| Implement Decision Tree Classifier. | Decision Tree Titanic Data Set.ipynb | Beth & Kalee |
| Logistic Regression Classification – ANN implementation | Logistic Regression - ANN - XGBoost Titanic Dataset.ipynb | Chris |
| SVM implementation. | SVM Prediction Titanic Dataset.ipynb | Beth |
| Wrote sections 1-4 of the report | Project Final Report Template.docx | All |