CS 589 Project Report

Introduction:

a. **Problem:** The Titanic dataset is commonly utilised in machine learning for a binary classification task: predicting whether a passenger survived or did not survive the Titanic disaster. This historical problem offers an opportunity to investigate the various factors that might have played a role in determining an individual's chances of survival.
b. **DataSet:** The dataset includes details about Titanic passengers, encompassing features like age, gender, class, ticket fare, and others. Each passenger is marked with a label indicating survival (1) or non-survival (0). Typically, the dataset is divided into a training set for constructing the model and a test set for assessing its effectiveness. The Titanic dataset comprises information on passengers, encompassing features such as age, gender, class, ticket fare, siblings/spouses aboard (SibSp), parents/children aboard (Parch), ticket number, cabin, and the port of embarkation (Embarked).
c. **Objectives:** The main objectives of working with the Titanic dataset are:
    i. **Exploratory Data Analysis and Cleaning:** Understand the dataset by exploring its characteristics and handling missing or inconsistent data.
    ii. **Feature Engineering:** Create new features or modify existing ones to improve the model's predictive performance.
    iii. **Model Training:** Utilise machine learning algorithms to train a model on the training set, using features to predict the survival outcome.
    iv. **Model Evaluation:** Evaluate the model's performance on the test set to assess its ability to generalise to new, unseen data.
    v. **Insights:** Gain insights into factors that influenced survival rates, which may include gender, age, class, or other features.

Methodology:

a. **Data preprocessing Steps:**
    i. **Handling Missing Values:** I have used the K-Nearest Neighbors (KNN) imputation technique to handle missing values in a DataFrame. Unlike traditional methods such as filling with the mean, KNN imputation estimates missing values by considering the values of their nearest neighbours. In this approach, each missing value is replaced with a value computed based on the values of its k-nearest neighbours, where 'k' is a user-specified parameter. This method takes into account the similarity between data points, potentially offering a more nuanced and context-aware imputation compared to simple statistical measures

like the mean. Also, for the Embarked column I have used mode for filling missing values as there are only 2 empty values.

ii. **Handling Outliers:** Addressing outliers is essential in data preprocessing to prevent extreme values from disproportionately influencing the model's performance. The interquartile range (IQR) is employed as a measure of statistical dispersion to identify and handle outliers. By calculating the IQR and determining the upper bridge values (3 times the IQR above the third quartile), the method focuses on capturing and neutralising extreme data points that could potentially introduce noise or bias to the model. This approach, based on the IQR, provides a robust way to identify and adjust for outliers, promoting a more resilient and accurate machine learning model.

iii. **Data Encoding (Categorical Variables):** Categorical variables undergo one-hot encoding through the **alternative_ohe** function. This method employs a customised approach, considering a specified percentage of value counts for each feature. The function iterates through the chosen variables, creating a DataFrame for each feature to determine value counts and value ratios. It selects categories that collectively cover the specified percentage of the data, generating binary columns for these selected categories. Finally, the original categorical columns are dropped, and the transformed DataFrame is returned, capturing a subset of categories based on the specified percentage. Continuous variables (Fare and Age) are binned into distinct intervals, followed by the application of one-hot encoding to represent these interval categories. The Columns to which one hot encoding is applied are: 'Embarked','Sex','Title', 'Age_bin', 'Fare_bin', 'FamilyGroup'.

iv. **Normalisation:** Normalisation is not explicitly applied in this case, as the continuous variables have been transformed into one-hot encoded binary columns. One-hot encoding inherently handles the scale of the variables, as it represents categorical information through binary values, rendering explicit normalisation unnecessary. This approach maintains the integrity of the categorical information while addressing the scale-related concerns associated with continuous variables.

b. Feature Engineering:

i. **Title Extraction:** Passenger titles are derived from the 'Name' column, offering additional information on social status. Less common titles are collectively grouped under the 'Other' category for simplification.

ii. **Age Feature:** Age column is binned into ['Children','Teenage','Adult','Elder'], providing a more generalised depiction of different age ranges.

iii. **Fare Feature:** Fare column is binned into ['Low_fare','median_fare','Average_fare','high_fare'], offering a more generalised representation of various fare ranges.

iv. **Family Features:** A new column named 'FamilySize' is generated by combining the values from the 'SibSp' (siblings/spouses) and 'Parch' (parents/children) columns. Subsequently, the 'FamilySize' column is categorised into groups, specifically [‘alone’, ‘small’, ‘large’], and the results are stored in a new column named 'FamilyGroup'.

v. **One Hot Encoding:** Converting all the categorical columns into binary columns.

c. **Implementation of three different learning approaches**

i. **Fix-shape universal approximators:**

1. **Model Choice:** Support Vector Classifier
2. **Classifier Choice:** Support Vector Machine is chosen for its effectiveness in handling high-dimensional data and its ability to find non-linear decision boundaries through the kernel trick. It is particularly useful when the decision boundary is complex and not easily linearly separable.
3. **Model Initialization:** The Support Vector Machine is initialised using SVC (Support Vector Classification).
4. **Hyperparameter Tuning:** Hyperparameter tuning is done through GridSearchCV with parameters such as **C**, **kernel**, and **gamma**. The grid search is performed in conjunction with k-fold cross-validation.
5. **Cross Validation:** K-fold cross-validation is used to assess the SVM model's performance

ii. **Neural network based approach:**

1. **Model Choice:** MLP Classifier
2. **Classifier Choice:** Multi-Layer Perceptron is chosen for its capability to capture complex relationships in data through its hidden layers and non-linear activation functions. It is a suitable choice for tasks where intricate patterns and interactions between features need to be learned.
3. **Model Initialization:** The Multi-Layer Perceptron is initialised using MLPClassifier.
4. **Hyperparameter Tuning:** Hyperparameter tuning is conducted using GridSearchCV with parameters such as **hidden_layer_sizes, activation, alpha, max_iter,** and **learning_rate**. The grid search is performed in conjunction with k-fold cross-validation.
5. **Cross Validation:** K-fold cross-validation is used to assess the MLP model's performance

iii. Tree-based approaches:

1. **Model Choice:** Random Forest Classifier (RFC)
2. **Classifier Choice:** Random Forest Classifier is chosen for its ability to handle non-linear relationships, feature importance estimation, and robustness against overfitting. It is an ensemble method that combines the predictions of multiple decision trees, making it suitable for complex datasets.
3. **Model Initialization:** The Random Forest Classifier is initialised by creating an instance of RandomForestClassifier.
4. **Hyperparameter Tuning:** Hyperparameter tuning is performed using GridSearchCV with parameters like **n_estimators, max_depth, min_samples_split**, and **min_samples_leaf**. Grid search is employed to find the combination of hyperparameters that maximises the accuracy on cross-validated data.
5. **Cross Validation:** K-fold cross-validation is used to assess the RFC model's performance

Validation Method and Configurations:

The implemented validation method involves k-fold cross-validation with k values [5,10,15] utilising the KFold function with shuffling and a random seed for reproducibility. This technique is applied during hyperparameter tuning using GridSearchCV for all approaches. The **scoring='accuracy'** parameter indicates the evaluation metric used to assess model performance during hyperparameter tuning. Overall, k-fold cross-validation is a well-established strategy, ensuring thorough model evaluation and hyperparameter optimization by systematically rotating through different training and validation subsets.

Results

a. Support Vector Classifier:

i. Hyperparameters rationale:

1. **C:** The regularisation parameter. A smaller C allows for a more flexible decision boundary, which is crucial when dealing with non-linear relationships. However, too small a C may lead to overfitting, so a balanced value is chosen.
2. **kernel and gamma:** These parameters define the type of decision boundary and the influence of individual training samples. The 'rbf' kernel is versatile and often works well for various problems. 'Gamma' defines how far the influence of a

single training example reaches; a small gamma results in a large influence, making the decision boundary more flexible.

ii. **Best Hyperparameter Values:** {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}

b. **MLP Classifier:**

i. **Hyperparameters Rationale:**

1. **hidden_layer_sizes:** The architecture of the neural network. The choice of a single hidden layer with 100 neurons is a simple yet effective configuration for this task. Adding more layers or neurons might increase complexity but could lead to overfitting given the dataset size.
2. **activation:** The 'relu' activation function is chosen for hidden layers due to its efficiency in handling non-linearities.
3. **alpha:** Regularisation term. A small value is chosen to add a small amount of regularisation and prevent overfitting.
4. **max_iter:** The maximum number of iterations for optimization. It ensures that the MLP converges to a solution within a reasonable number of iterations.
5. **learning_rate:** The adaptive learning rate allows for efficient convergence, and 'adaptive' ensures that the learning rate adapts during training.

ii. **Best Hyperparameter Values:** {'activation': 'relu', 'alpha': 0.1, 'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'max_iter': 300}

c. **Random Forest Classifier:**

i. **Hyperparameters Rationale:**

1. **n_estimators**: The number of trees in the forest. A higher number may improve the model's performance, but it comes with increased computational cost. A balance is struck to ensure an adequate number for capturing variability without excessive computation.
2. **max_depth:** The maximum depth of the trees. Limiting the depth helps prevent overfitting by restricting the complexity of individual trees.
3. **min_samples_split and min_samples_leaf:** These parameters control the size of the nodes. Smaller values allow the model to capture fine details, but too small may lead to overfitting. Hence, they are chosen to strike a balance.

ii. **Best Hyperparameters Values:** {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}

Choice of Metrics:

The primary metric selected for assessing model performance is accuracy, offering a holistic measure of correct predictions. In addition to accuracy, precision, recall, and F1-score are considered crucial supplementary metrics, especially in imbalanced datasets. Precision provides insight into the accuracy of positive predictions, while recall assesses the model's ability to capture all positive instances. F1-score, as a harmonized metric, balances precision and recall and proves valuable in scenarios with uneven class distributions. The inclusion of these metrics ensures a nuanced evaluation, vital for understanding the model's strengths and areas for improvement, particularly in situations where class imbalances may impact traditional accuracy measurements. This comprehensive metric selection strategy aims to enhance the model's generalisation and reliability across diverse real-world scenarios.

Models Performance:

Validation Accuracy using SVC Classifier: **83.28%**
Validation Accuracy using MLP Classifier: **83.72%**
Validation Accuracy using Random Forest Classifier: **84.06%**

Training Accuracy using SVC Classifier: **83.27%**
Training Accuracy using MLP Classifier: **87.31%**
Training Accuracy using Random Forest Classifier: **87.09%**

The predicted values for each approach are uploaded to the kaggle to get the accuracies on the test data.

Accuracy using SVC Classifier: **77.51%**
Accuracy using MLP Classifier: **76.31%**
Accuracy using Random Forest Classifier: **76.79%**

Visualisations:

Classification Report: The classification report provides precision, recall, and F1-score for each class, offering a comprehensive evaluation of the model's performance on different aspects of classification.

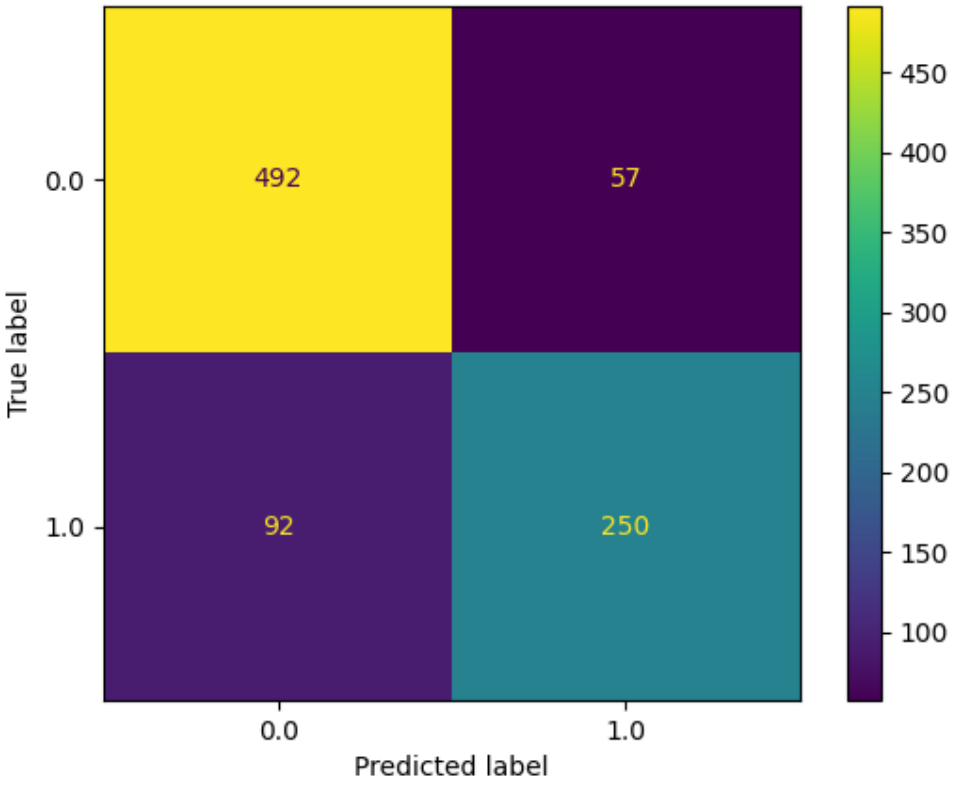
Confusion Matrix: The confusion matrix visually summarizes the performance of a classification model by displaying the counts of true positive, true negative, false positive, and false negative predictions.

SVC Classifier:

Classification Report

Label	Precision	Recall	F1 Score
0.0	0.84	0.90	0.87
1.0	0.81	0.73	0.77

Confusion Matrix

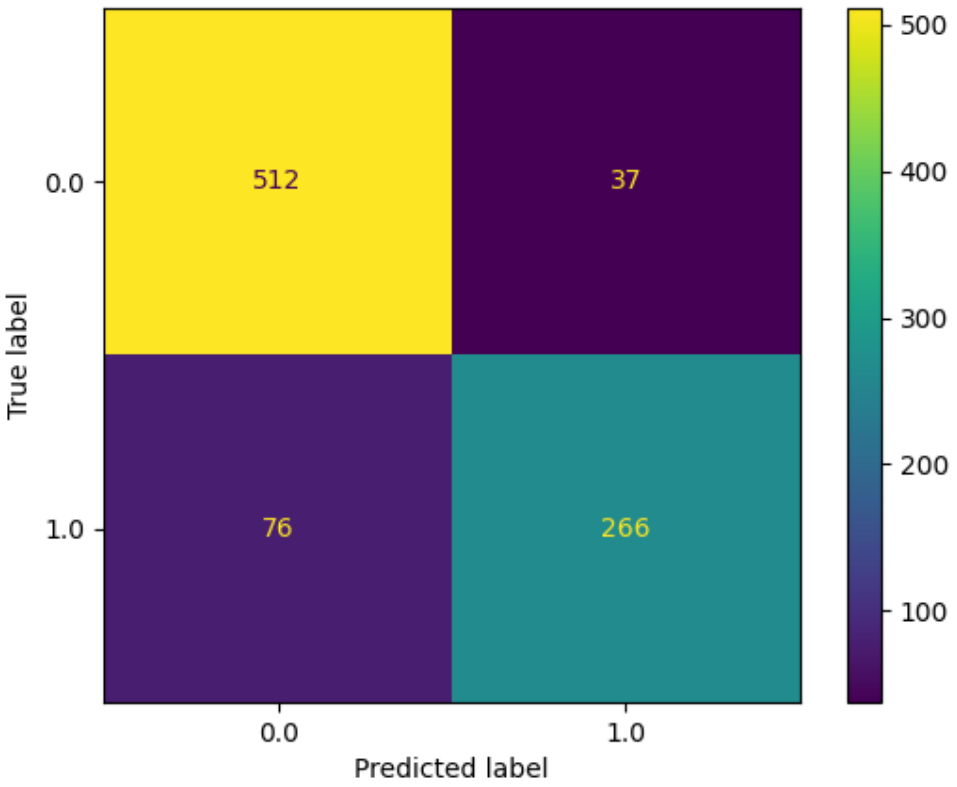


MLP Classifier

Classification Report

Label	Precision	Recall	F1 Score
0.0	0.87	0.93	0.9
1.0	0.88	0.78	0.82

Confusion Matrix

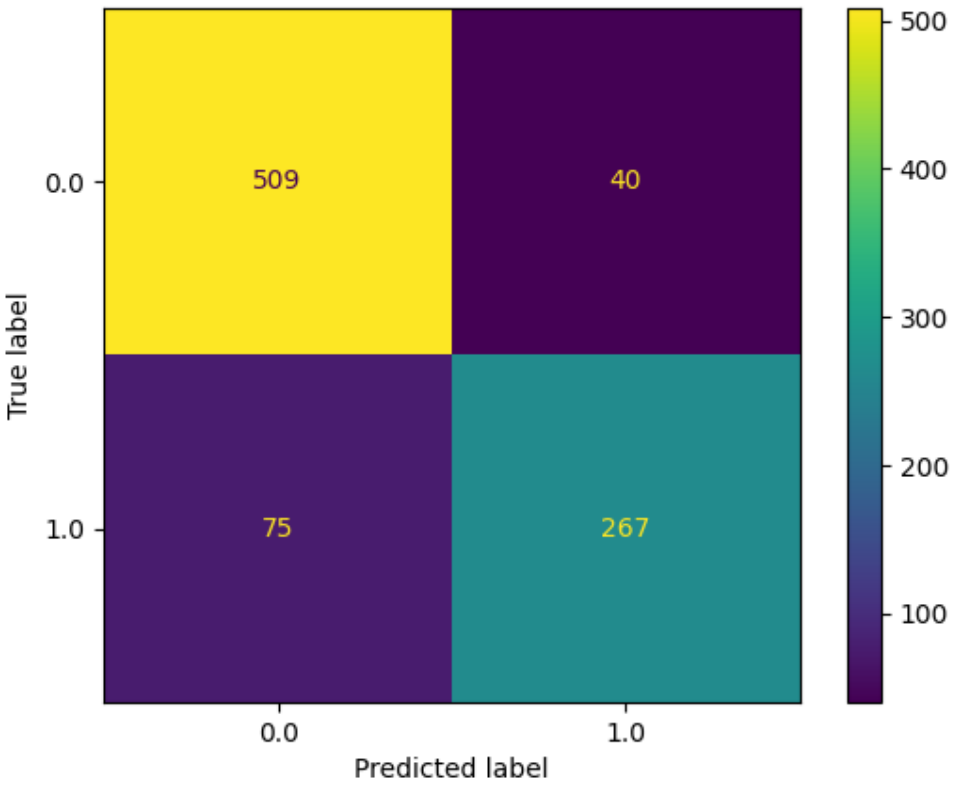


Random Forest Classifier:

Classification Report

Label	Precision	Recall	F1 Score
0.0	0.87	0.93	0.9
1.0	0.87	0.78	0.82

Confusion Matrix



Accuracy Comparison Table

	SVC Classifier	MLP Classifier	Random Forest Classifier
Validation Accuracy	83.28	83.72	84.06
Training Accuracy	83.27	87.31	87.09
Test Accuracy	77.51	76.31	76.79

Conclusion:

In summary, the analysis of the Titanic dataset involved comprehensive data preprocessing, feature engineering, and the implementation of three distinct machine learning approaches: Support Vector Classifier (SVC), Multi-Layer Perceptron (MLP) Classifier, and Random Forest Classifier (RFC). The models were evaluated using various metrics, with RFC exhibiting the highest validation accuracy at 84.06%, followed closely by SVC at 83.28% and MLP at 83.72%. However, when applied to the test data, SVC achieved an accuracy of 77.51%, while RFC and MLP achieved accuracies of 76.79% and 76.31%, respectively.

Key findings include the effectiveness of the chosen models in predicting survival outcomes, with SVC, RFC and MLP demonstrating robust performance. The feature engineering strategies, such as title extraction, age and fare binning, and family size categorization, contributed to enhancing model performance. Insights into hyperparameter choices revealed the importance of balancing model complexity and avoiding overfitting. Notably, precision, recall, and F1-score were considered in addition to accuracy, providing a nuanced evaluation, particularly in imbalanced datasets.

Areas for improvement could involve further exploration of feature interactions, experimenting with different machine learning algorithms, and conducting a deeper analysis of misclassifications. Additionally, insights into the impact of specific features on model predictions could be valuable for refining the model. Consideration of ensemble methods or advanced techniques like gradient boosting may also offer opportunities to boost predictive accuracy. Overall, this project lays a solid foundation for survival prediction on the Titanic dataset, with potential for refinement and optimization in future iterations.