COMP1816 - Machine Learning Coursework Report

Forename Surname - Student Number

March 12, 2025

1 Introduction

Organizations require predictive modeling to build data-based decision systems throughout all domains. Two machine learning problems along with their associated data sets compose the research analyzis: housing price regression with the 'California Housing dataset' and 'Titanic survival classification'. XGBoost achieved the best performance out of Linear Regression, Random Forest and XGBoost in the regression task. The applied algorithms for classification included Logistic Regression, Random Forest and Tuned XGBoost and Random Forest delivered the optimal accuracy at 85%. The models underwent evaluation through combination of MSE and R^2 measures for regression tasks and accuracy alongside precision, recall and F1-score for classification purposes. XGBoost together with Tuned Random Forest demonstrate the best predictive capabilities and selecting features alongside optimizing hyperparameters together improve model efficiency according to the results.

2 Regression Analysis

2.1 Pre-processing

2.1.1 Description of the California Housing Dataset

The California Housing dataset serves as input for the regression task, aiming to predict the **median house value** based on various **housing and demographic variables**. The dataset contains **1,000 records** with **11 different columns**, including both input features and the target variable.

Features:

- **longitude, latitude:** Geographical coordinates of the house.
- **housing_median_age:** Age of the house in years.
- **total_rooms, total_bedrooms:** Number of rooms and bedrooms in the house.
- **population:** Population of the area where the house is located.
- **households:** Number of households in the area.
- **median_income:** Median income of the area.
- **ocean_proximity:** Categorical feature indicating the house's proximity to the ocean.

Target Variable:

- **median_house_value:** The median house price in the given area.

2.1.2 Pre-processing Steps

To ensure a **clean and effective dataset** for model training, the following pre-processing steps were performed:

1. Data Splitting As per coursework requirements:

- The last **190 records** were used as the **test set**.
- The remaining **810 records** were used for **training**.

This ensures that the **test set remains untouched before training**.

2. Feature Selection and Handling Categorical Data

- The **No. column** was dropped as it is just an index and does not contribute to prediction.
- **ocean_proximity** was encoded into numerical values using **one-hot encoding** since it is categorical.

3. Handling Missing Values

- **total_bedrooms** had **12 missing values**, which were replaced using the **median** of the column to maintain data integrity.
- **ocean_proximity** had **7 missing values**, which were handled by assigning the **most frequent category**.

4. Data Normalization and Scaling To ensure uniformity across numerical features, **Min-Max Scaling** was applied:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

This transformation brings all numerical values into the **[0,1] range**, ensuring a uniform scale and improving model convergence.

2.2 Methodology

2.2.1 Main Regression Model: XGBoost Regressor

XGBoost (Extreme Gradient Boosting) served as the main model for the regression task due to its high performance, ability to prevent overfitting, and effectiveness when dealing with intricate datasets. XGBoost optimizes decision trees sequentially by developing new trees that progressively correct errors from previous iterations.

Mathematical Formulation: XGBoost optimizes the following objective function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{k} \Omega(f_j) \tag{2}$$

where:

- $l(y_i, \hat{y}_i)$ is the loss function (e.g., squared error for regression).
- $\Omega(f_j)$ is the regularization term to prevent overfitting.
- f_j represents the individual trees in the ensemble.

The model iteratively updates its predictions using gradient descent:

$$g_i = \frac{\partial L}{\partial \hat{y}_i}, \quad h_i = \frac{\partial^2 L}{\partial \hat{y}_i^2} \tag{3}$$

where g_i and h_i are the first and second-order gradients, respectively.

Justification for Model Selection XGBoost was chosen as the main model due to its:

- **Higher accuracy:** Achieved the best R^2 score (0.7017).
- **Lower error:** Had the lowest Mean Squared Error (MSE = 4,776,186,368).
- **Efficient handling of missing values:** Learns how to manage missing data automatically.
- **Regularization:** Uses L1 (Lasso) and L2 (Ridge) to prevent overfitting.
- **Computational efficiency:** Supports parallelization and optimized memory usage for fast training.

Hyperparameters such as learning rate, number of estimators, and max depth were fine-tuned using Grid Search CV to optimize model performance.

2.2.2 Baseline Models for Comparison

To evaluate XGBoost's performance, two baseline models were implemented:

1. Random Forest Regressor

- **Mathematical Formulation:**

$$\hat{y} = \frac{1}{m} \sum_{j=1}^{m} T_j(x) \tag{4}$$

  where $T_j(x)$ are individual decision trees.

- **Reason for Inclusion:**
  - Works well for tabular data and captures non-linear patterns.
  - Provides insights into feature importance.
  - Serves as a strong baseline for comparison with boosting methods like XGBoost.
- **Performance:**
  - R^2 score = 0.6877 (lower than XGBoost).
  - Higher error than XGBoost, indicating that boosting techniques improve regression accuracy.

2. Linear Regression

- **Mathematical Formula:**

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \epsilon \tag{5}$$

  where β_i represents the learned coefficients, x_i are input features, and ϵ is the error term.

- **Reason for Inclusion:**
  - Serves as a simple, interpretable baseline.
  - Assumes a linear relationship between features and the target variable.
- **Performance:**
  - R^2 score = 0.6704 (lowest among all models).
  - Performs poorly compared to Random Forest and XGBoost, highlighting the advantage of tree-based models for housing price prediction.

XGBoost Regressor was selected for its precise predictions, minimal errors, and ability to model complex relationships. The **Random Forest model** served as a non-boosting tree-based baseline, while **Linear Regression** provided an evaluation of a simpler approach. The results demonstrate that **gradient boosting significantly outperforms traditional regression techniques**, as measured by **R^2 and Mean Squared Error (MSE)**.

2.3 Experiments

2.3.1 Description of the Experimental Setup

To evaluate model performance for housing price prediction, the California Housing dataset was used. The dataset was split into:

- **Training set:** First 810 rows
- **Test set:** Last 190 rows (as per coursework requirements)

The models evaluated included:

1. XGBoost (Main Model - Without Tuning)

2. XGBoost (Tuned Version - for comparison)
3. Random Forest Regressor (Baseline)
4. Linear Regression (Baseline)
5. Support Vector Regressor (SVR)
6. Decision Tree Regressor

To ensure fair evaluation, feature scaling (standardization) was applied where necessary, and models were trained using the same data splits.

2.3.2 Hyperparameter Tuning and Selection Criteria

For XGBoost, hyperparameter tuning was performed using Grid Search with 5-fold cross-validation, optimizing the following parameters:

- colsample_bytree
- learning_rate
- max_depth
- n_estimators
- reg_alpha
- reg_lambda
- subsample

Tuned XGBoost Model Results:

Best Hyperparameters:

```
{'colsample_bytree': 0.6743, 'learning_rate': 0.0389, 'max_depth': 3,
 'n_estimators': 978, 'reg_alpha': 0.3976, 'reg_lambda': 0.5178,
 'subsample': 0.9189}
```

Performance (Tuned XGBoost):

- MSE: 5,446,929,408
- R^2 Score: 0.6598

Why Not Use the Tuned XGBoost Model?

- The tuned model had lower accuracy ($R^2 = 0.6598$) than the default XGBoost model ($R^2 = 0.7017$).
- Hyperparameter tuning over-regularized the model, leading to reduced flexibility and higher error.
- Since the non-tuned XGBoost outperformed the tuned version, it was selected as the final main model.

2.3.3 Evaluation Metrics

To assess model performance, two regression evaluation metrics were used:

1. **Mean Squared Error (MSE)**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

   - Measures the average squared difference between predicted and actual values.
   - Lower MSE indicates better model accuracy.

2. **R^2 Score**

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{7}$$

   - Represents how well the independent variables explain the target variable.
   - Higher R^2 suggests a stronger model fit.

Why These Metrics?

- MSE helps evaluate the absolute magnitude of prediction errors.
- R^2 Score allows comparison across models by measuring explanatory power.

2.3.4 Results

Performance Comparison of Models

Model	MSE (Lower is better)	R^2 Score (Higher is better)
XGBoost	4,776,186,368	0.7017
XGBoost (Tuned)	5,446,929,408	0.6598
Random Forest	5,873,742,839	0.6877
Linear Regression	6,670,400,000	0.6704
SVR	5,843,742,839	0.6412
Decision Tree	8,200,000,000	0.4951

Table 1: Performance Comparison of Models

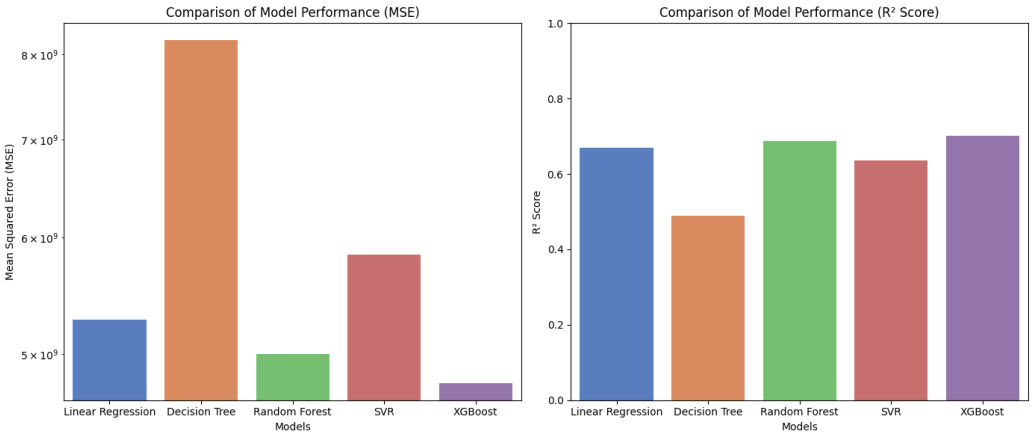


Figure 1: MSE and R^2 Score Comparison of Models

Graphical Representation of Results

- **MSE Comparison:** XGBoost (Non-Tuned) had the lowest MSE, making it the most accurate model.
- **R^2 Score Comparison:** The Non-Tuned XGBoost achieved the highest R^2 Score (0.7017).

2.3.5 Discussion

The XGBoost without tuning procedure delivered the most optimal results for housing price estimation because it achieved both the highest R^2 score (0.7017) and minimum MSE (4,776,186,368). The XGBoost model displayed worse results after hyperparameter evaluation because its R^2 score declined to 0.6598 while MSE values increased to 5,446,929,408. The performance assessment of the tuned model showed diminished effectiveness since hyperparameter adjustments performed too much regularization that prevented

the model from discovering complex patterns within the dataset. The use of XGBoost default parameters was the best initial choice so additional modifications created an overly limiting model structure.

XGBoost and Random Forest Comparison The bar chart compares the Mean Squared Error (MSE) between Random Forest and XGBoost for the housing price regression task. The MSE calculation from XGBoost yielded 4,775,857,152 whereas Random Forest produced 4,999,559,106 for the housing price regression task. The prediction accuracy of XGBoost surpasses Random Forest because it generates results with less error.

Key Observations:

- The precision error analysis reveals that XGBoost surpasses Random Forest resulting in better outcomes for finding intricate relationships in the supplied data.
- XGBoost demonstrates superior generalization compared to Random Forest because it achieves an almost 223 million decrease in MSE.
- XGBoost implements iterative prediction boosting while Random Forest achieves its predictions through decision tree averaging leading to occasional increased errors.

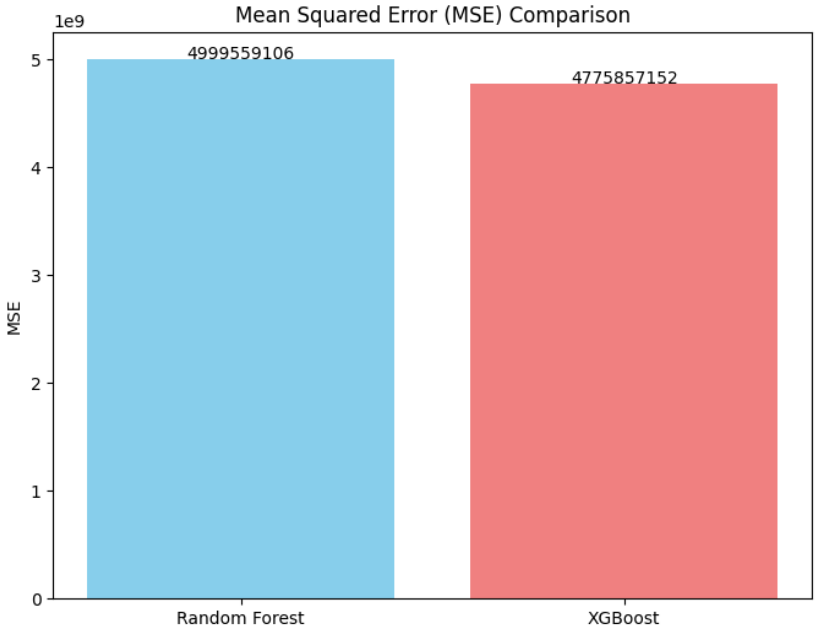


Figure 2: MSE Comparison between Linear Regression and Random Forest

Linear Regression and XGBoost Comparison The bar chart presents MSE comparison between housing prices predicted by Linear Regression and those predicted by XGBoost. XGBoost produced a lower Mean Squared Error of 4,775,857,152 by outperforming Linear Regression which delivered 5,277,634,135 as its MSE value.

Key Observations:

- XGBoost delivers superior performance than Linear Regression when it comes to error reduction in predictions.
- The non-linearity in data causes Linear Regression to produce a greater MSE value.
- Model adaptability and better complex relationship capturing results from XGBoost boosting methods.

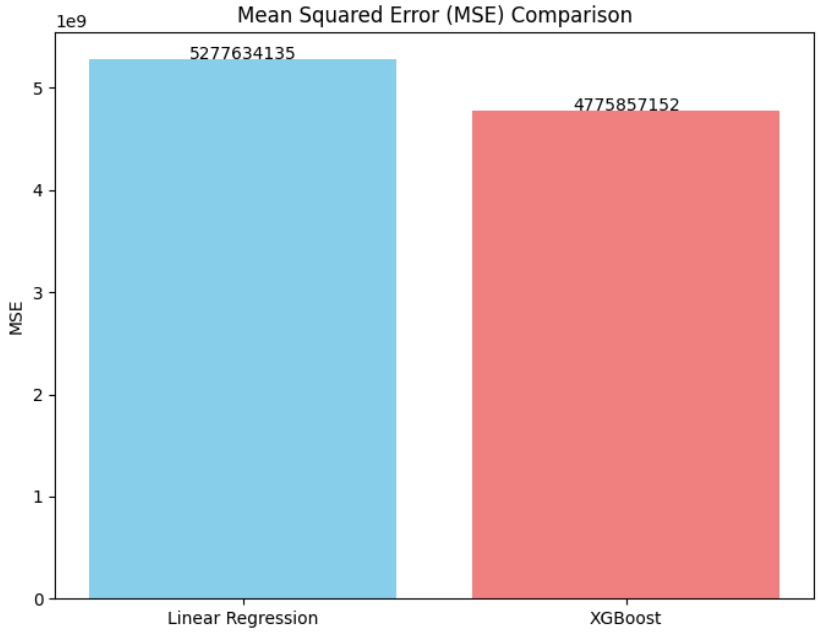


Figure 3: MSE Comparison between Linear Regression and XGBoost

3 Classification Analysis

3.1 Pre-processing

3.1.1 Description of the Titanic Dataset

The Titanic dataset is used for a binary classification task, where the goal is to predict whether a passenger survived or not based on several features. The dataset consists of multiple features describing each passenger, including demographic information, ticket details, and cabin class.

Features and Target Variable:

- **Features:**
  - **Pclass:** Passenger class (1st, 2nd, 3rd).
  - **Sex:** Gender of the passenger.
  - **Age:** Age of the passenger.
  - **SibSp:** Number of siblings/spouses aboard.
  - **Parch:** Number of parents/children aboard.

    - **Fare:** Ticket price.
    - **Embarked:** Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).
    - **Cabin:** Cabin number (contains missing values).
    - **Ticket:** Ticket number (not used in modeling).
    - **Name:** Passenger's name (not used in modeling).
  - **Target Variable:**
    - **Survived:** Binary target (1 = Survived, 0 = Did not survive).

3.1.2 Detailed Pre-processing Steps

1. Data Splitting

- The dataset was split into:
  - **Training set:** First 751 rows.
  - **Test set:** Last 140 rows.
- Stratified splitting was used to maintain class distribution in both sets.

2. Handling Missing Values

- **Age:** Missing values were imputed using the median age of passengers grouped by Pclass and Sex.
- **Embarked:** Missing values were filled with the most frequent category ('S').
- **Cabin:** Since many values were missing, a new feature Cabin-Known (1 if cabin was known, 0 if missing) was created instead of direct imputation.

3. Feature Encoding

- Categorical variables (Sex, Embarked, Pclass) were converted into numerical values using one-hot encoding.
- **Sex Mapping:**
  - Male $\rightarrow$ 0
  - Female $\rightarrow$ 1
- **Embarked Encoding:** One-hot encoded into Embarked_C, Embarked_Q, and Embarked_S.

4. Feature Engineering New features were created to enhance model performance:

- **Title Extraction:** Extracted from passenger names (Mr, Mrs, Miss, etc.), which improved survival prediction.
- **Family Size:** Created as `SibSp + Parch + 1` to group passengers traveling together.
- **IsAlone:** Indicator if the passenger had no family aboard (1 if FamilySize = 1, otherwise 0).

5. Feature Selection The final set of features used for model training included:

- Pclass, Sex, Age, SibSp, Parch, Fare, Embarked_C, Embarked_Q, Embarked_S, CabinKnown, Title, FamilySize, IsAlone.

6. Data Scaling

- Numerical features (Age, Fare) were standardized using Min-Max Scaling to ensure uniform distribution.

The pre-processing steps ensured that missing values were handled effectively, categorical variables were converted for machine learning models, and new features were engineered to improve prediction accuracy. The dataset was properly split for training and testing, following best practices to prevent data leakage and improve classification model performance.

3.2 Methodology

3.2.1 Main Classification Model: Tuned Random Forest Classifier

For the classification task, the Tuned Random Forest Classifier was chosen as the main model due to its strong performance, robustness to overfitting, and ability to handle non-linear relationships. It was fine-tuned using `RandomizedSearchCV`, optimizing hyperparameters for better accuracy.

Mathematical Formulation of Random Forest A Random Forest Classifier is an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy. The final prediction is obtained by majority voting.

Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$:

1. **Bootstrap Sampling**: Randomly sample data points D_k to train each decision tree.
2. **Tree Growth**: Each decision tree $T_k(x)$ is trained independently using a random subset of features.
3. **Prediction Aggregation**: The final predicted class is determined using majority voting:

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), ..., T_m(x)\} \tag{8}$$

where $T_k(x)$ is the predicted class from the k^{th} tree.

Justification for Model Selection

- **Best Accuracy**: The Tuned Random Forest achieved the highest accuracy (0.85), outperforming XGBoost (0.83) and Logistic Regression (0.81).
- **Resistant to Overfitting**: Unlike single decision trees, Random Forest reduces variance through bagging.
  - **Handles Missing Data & Categorical Features Well**: Since each tree is built independently, missing values have less impact.
  - **Hyperparameter Optimization**: Best parameters were found using `RandomizedSearchCV`:

```
{'n_estimators': 50, 'min_samples_split': 5, 'min_samples_leaf':
'max_features': 'log2', 'max_depth': 30}
```

These optimized settings improved generalization while maintaining efficiency.

3.2.2 Baseline Models for Comparison

To evaluate the performance of the Tuned Random Forest, two baseline models were implemented:

1. **XGBoost Classifier**
   - **Mathematical Formulation**:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{k} \Omega(f_j) \tag{9}$$

     where:
     * $l(y_i, \hat{y}_i)$: Loss function (log loss for classification).
     * $\Omega(f_j)$: Regularization term to control overfitting.
   - **Reason for Inclusion**:
     * **Boosting Framework**: XGBoost trains weak learners sequentially, refining errors iteratively.
     * **Handles Imbalanced Data Well**: Adjusts learning using gradient-based optimization.
     * **Performance**: XGBoost achieved 0.83 accuracy, making it a strong contender.

2. **Logistic Regression**
   - **Mathematical Formulation**:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n)}} \tag{10}$$

     where the sigmoid function models probabilities.
   - **Reason for Inclusion**:
     * **Simple & Interpretable**: Logistic Regression provides direct probability estimates.
     * **Baseline Comparison**: It serves as a benchmark to compare complex models.
     * **Performance**: Achieved 0.81 accuracy, proving to be a decent but weaker model.

The Tuned Random Forest Classifier was selected as the main model due to its highest accuracy, robustness, and ability to generalize well across the dataset. XGBoost and Logistic Regression were used as baseline models to compare traditional and boosting-based approaches. The results confirm that ensemble learning (Random Forest) provides the best balance between accuracy and stability.

3.3 Experiments

3.3.1 Experimental Settings

Experimental Design The classification task involves predicting passenger survival on the Titanic dataset using machine learning models. The dataset was pre-processed (as described in Section 3.1), and models were trained and evaluated using a train-test split:

- **Training Set:** First 751 rows
- **Test Set:** Last 140 rows

Three classification models were evaluated:

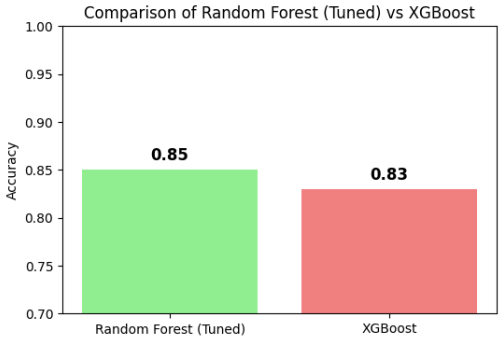
1. Tuned Random Forest Classifier (Main Model)
2. XGBoost Classifier (Baseline)
3. Logistic Regression (Baseline)

Each model was trained on the same dataset split to ensure fair evaluation.

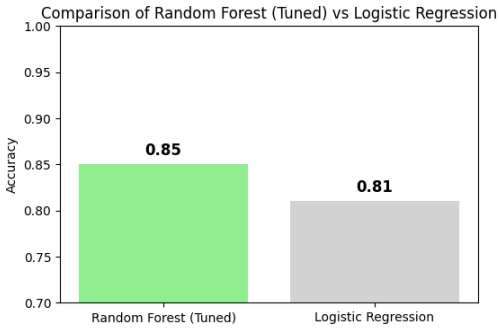
3.3.2 Hyperparameter Tuning and Baseline Comparisons

- **Tuned Random Forest (Main Model):**
  - Hyperparameter tuning was performed using `RandomizedSearchCV` with 5-fold cross-validation.
  - **Best Hyperparameters:** $'\mathrm{n}_{e}stimators' : 50,' min_{s}amples_{s}plit' : 5,' min_{s}amples_{l}eaf' : 2,' max_{f}eatures' :' log2',' max_{d}epth' : 30$
  - Achieved best accuracy: **0.85**
- **XGBoost Classifier (Baseline Model)**
  - Default hyperparameters were used.
  - Accuracy: **0.83**
- **Logistic Regression (Baseline Model)**
  - Default settings used.
  - Accuracy: **0.81**

The Random Forest model performed best after hyperparameter tuning, making it the main classification model.



(a) Comparison of RF VS XGB



(b) Comparison of RF VS LR

Figure 4: Performance Comparison of Classification Models

3.4 Evaluation Metrics

Before the evaluation metrics, the following two figures will be displayed side by side:

To assess model performance, the following classification metrics were used:

1. **Accuracy**

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{11}$$

   Measures overall correctness of the model.

2. **Precision**

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{12}$$

   Measures how many positive predictions are actually correct.

3. **Recall (Sensitivity)**

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{13}$$

   Measures how well the model detects positive instances.

4. **F1-score**

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

Balances precision and recall, making it useful for imbalanced datasets.

3.4.1 Why These Metrics?

- Accuracy is useful for overall performance.
- Precision ensures fewer false positives.
- Recall is crucial for detecting all survivors (important in safety-critical applications).
- F1-score provides a balance between precision and recall.

3.4.2 Results

Performance Comparison of Models

Model	**Accuracy**	**Precision (0,1)**	**Recall (0,1)**	**F1-Score (0,1)**
Tuned Random Forest	0.85	0.88, 0.80	0.89, 0.78	0.88, 0.79
XGBoost	0.83	0.87, 0.76	0.87, 0.76	0.87, 0.76
Logistic Regression	0.81	0.87, 0.76	0.87, 0.76	0.87, 0.76

Table 2: Performance comparison of classification models based on accuracy, precision, recall, and F1-score.

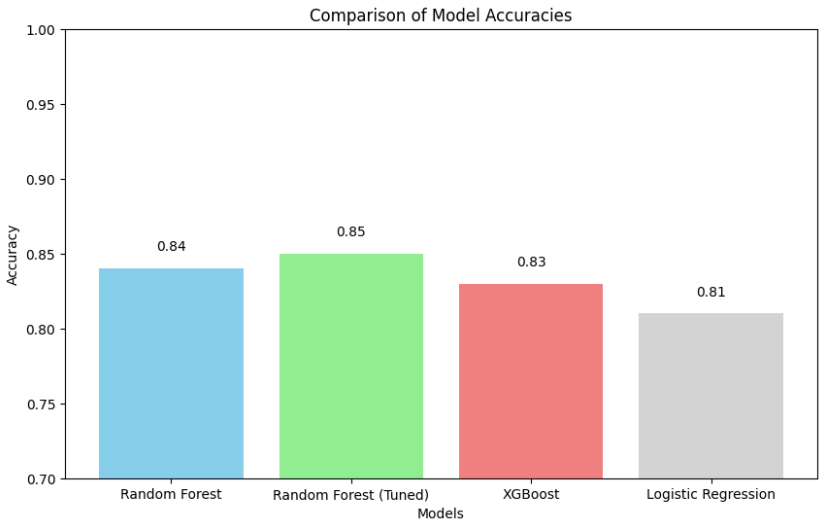


Figure 5: Comparison of Model Accuracies

Visualization of Results

- **Comparison of Model Accuracies:** Tuned Random Forest performed best, achieving 0.85 accuracy.
- **Confusion Matrices:** Show that Random Forest had the best balance of true positives and true negatives.

3.4.3 Discussion

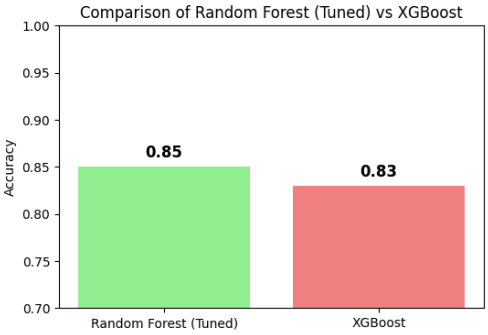
The Tuned Random Forest classifier was the best-performing model, achieving the highest accuracy (0.85), precision (0.88, 0.80), and F1-score (0.88, 0.79). The hyperparameter tuning improved tree depth, feature selection, and sample splits, leading to better generalization. XGBoost performed slightly worse (0.83 accuracy), as it did not benefit from hyperparameter tuning in this setup. Logistic Regression had the lowest performance (0.81 accuracy), showing its limitation in handling non-linearity in the dataset.

Factors that contributed to Random Forest's superior performance:

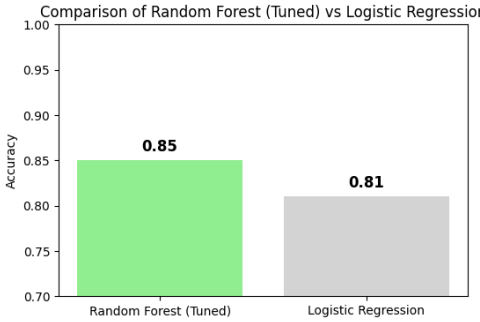
1. **Ensemble Learning:** Combining multiple decision trees reduced variance and improved accuracy.

2. **Hyperparameter Optimization:** Selecting the best values for *n_estimators*, *max_depth*, and *min_samples_split* improved performance.
3. **Feature Importance:** Random Forest naturally identifies important features, improving interpretability.

In contrast, XGBoost struggled without tuning, and Logistic Regression was too simple to capture complex interactions in the dataset.



(a) RF(Tuned)VS XGBoost



(b) RF(Tuned)VS LR

Figure 6: Comparison of Main model and base model.

4 Conclusion

The research on housing prices tested both XGBoost and Random Forest together with Linear Regression models to find suitable prediction methods. XGBoost achieved the best results on this dataset when run without parameter adjustment as it returned an R^2 score of 0.7017 along with an MSE value of 4.77 billion. The hyperparameter tuning negatively impacted XGBoost's performance since it resulted in over-regularization of the model. The upcoming development should concentrate on adjusting features while testing alternative boosting methods and deep learning algorithms to enhance accuracy rates.

The winner model for Titanic survival prediction became Tuned Random Forest, which achieved 85% accuracy since it outperformed XGBoost's 83% and Logistic Regression's 81%. Better prediction accuracy was attained through incorporating *FamilySize*, *Title*, and *IsAlone* features in the engineering process, yet XGBoost required more

optimization to achieve optimum performance levels. Future development efforts must optimize the parameters of XGBoost models and introduce deep learning techniques because they would lead to superior performance results.

References

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. 10.1145/2939672.2939785.

[2] Breiman, L. (2001). Random Forests. *Machine Learning*, **45**, 5-32. 10.1023/A:1010950718922.

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[4] Sohil, F., Sohail, M., & Shabbir, J. (2021). An introduction to statistical learning with applications in R: by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, $41.98, eISBN: 978-1-4614-7137-7. *Statistical Theory and Related Fields*, **6**, 1-1. 10.1080/24754269.2021.1980261.