ConfigurationManual

MScResearchProject
MScinArtificialIntelligence

Bhagyashree M Kenche
StudentID:x22228233

SchoolofComputing
NationalCollegeofIreland

Supervisor: Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Bhagyashree M Kenche
Student ID:	x22228233
Programme:	MSc in Artificial Intelligence
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Rejwanul Haque
Submission Due Date:	2/09/2024
Project Title:	Machine Learning Models implemented into GUI Application for Accurate Prediction of Health Insurance Charges.
Word Count:	645
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Bhagyashree M Kenche
Date:	2nd September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Bhagyashree M Kenche
x22228233

1 Introduction

This Configuration Manual intends to present an complete explanation over the significant steps taken minutely into consideration for the success of this research paper " Prediction of Health Insurance Cost". The data covered below is segregated under respective sections namely, System Configuration, Software and Hardware details, Implementation with development and deployment process carried out to get the expected output from the code run.

2 System and Software Requirements

The Project was developed and implemented on the below configuration:

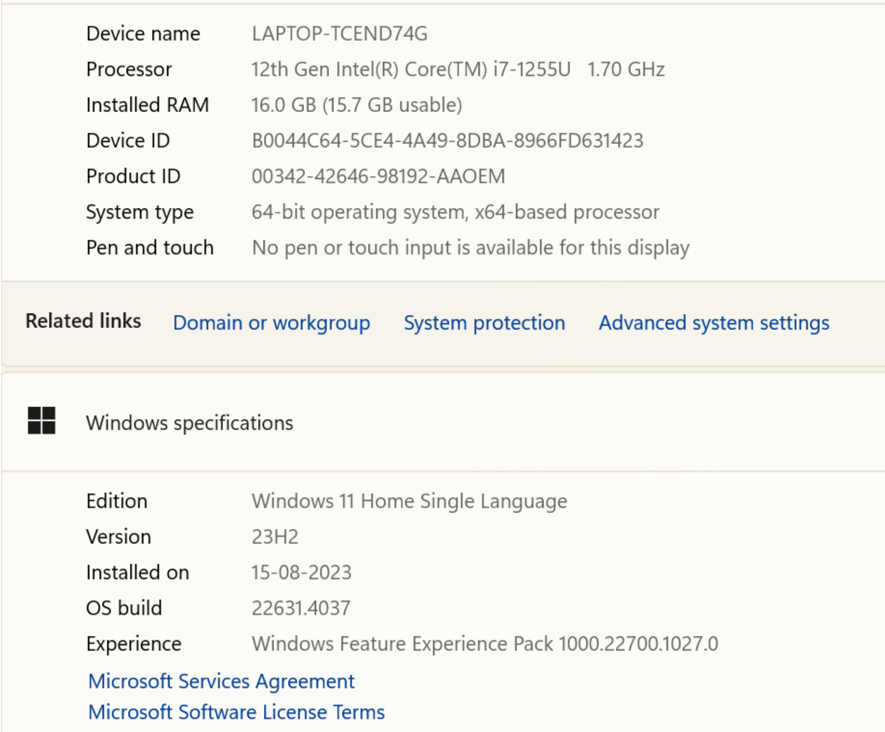

Figure 1: System Configuration.

Operating System	Windows 11
RAM	56.9 GB (Google Colab)
Disk Space	201.23 GB (Google Colab)

Runtime Model Name	12th Gen Intel(R) Core(TM) i7-1255U 1.70 GHz

Table 1: Hardware Configuration.

2.1 Software Requirements:

1. **Programming Language:** Python 3.11.4
2. **IDE:** Visual Studio

3 Python Libraries:

I used the following python libraries to conduct my research project of predicting melanoma skin cancer:

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. Tkinter
6. joblib
7. Sklearn

4 Dataset Description:

• Note: -Removed patient demographics in accordance with the GDPR rule.

The dataset which is used in this research project is open source that means publicly available on kaggle Harish Kumar DataLab. (2023). Medical Insurance Price Prediction Dataset. Kaggle.

4.1 Description:

This dataset consists of 2,772 records across 7 columns, with various attributes, such as, ages (18-64yrs), bmi, sex, smoker and so on.

The dataset is only for educational and non-commercial use. It is entirely synthetic and does not contain real patient data.

Size and Structure:

- **Total Records**: 2,772 entries
- **Total Features:** 7
- **Data Types:**

  Numerical: age, bmi and children

  Categorical: sex, smoker and region

Data Analysis and Visualization:

4.2 Data Distribution:

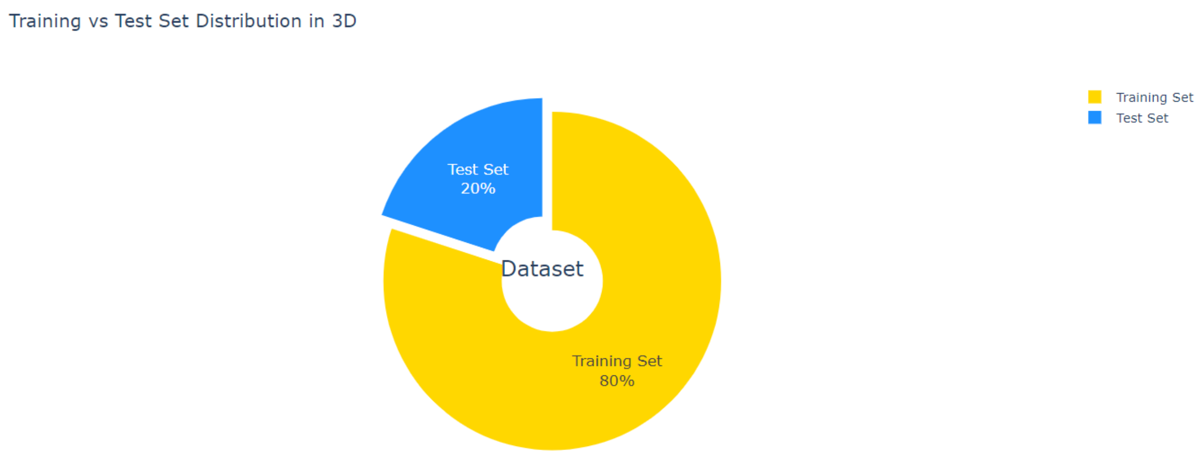

Figure 2: Data Distribution.

Age Distribution: The patient's age ranges from 18 to 64 years, with a mean age of approximately 39.1 years.

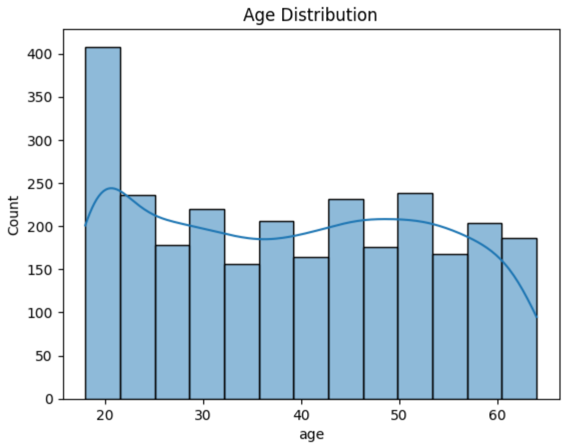

Figure 3: Age Distribution.

Sex Distribution: This consists on categorical data with values male or female..

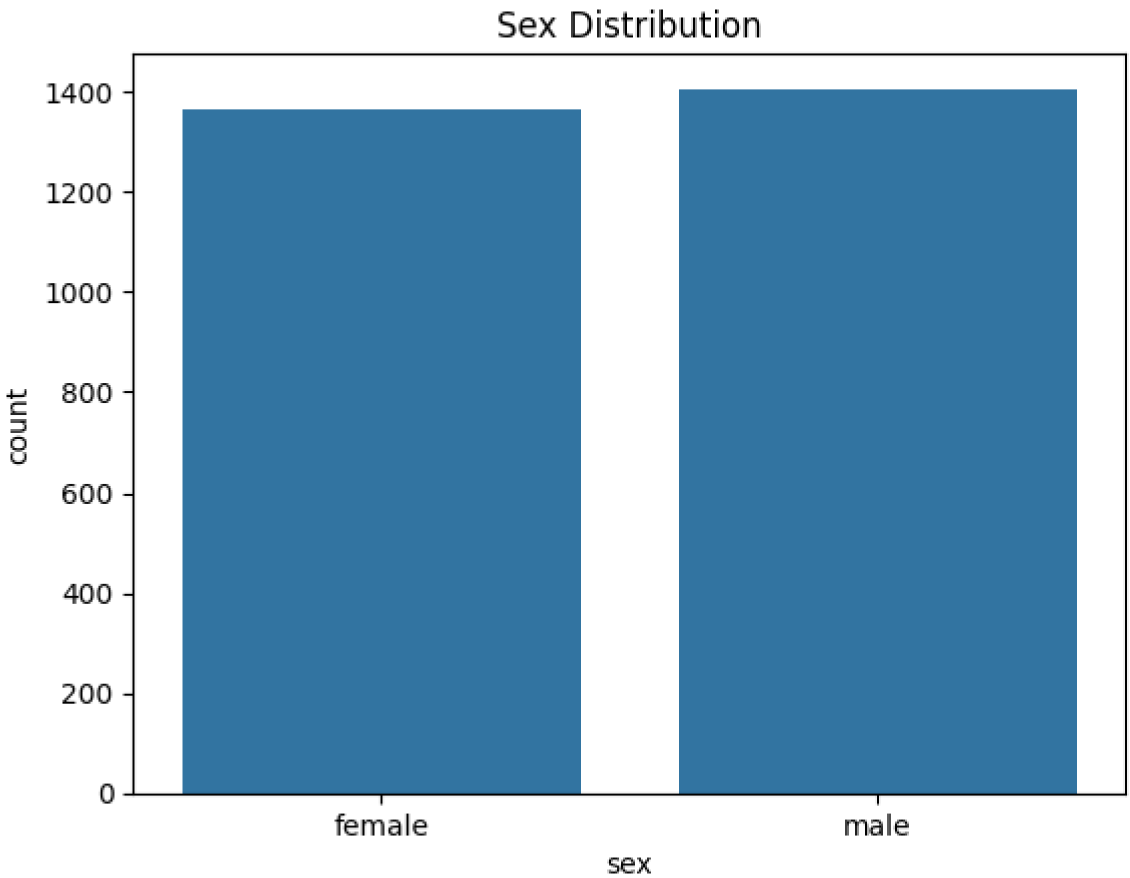

Figure 4: Sex Distribution.

BMI Distribution: This is a continuous numerical attribute.

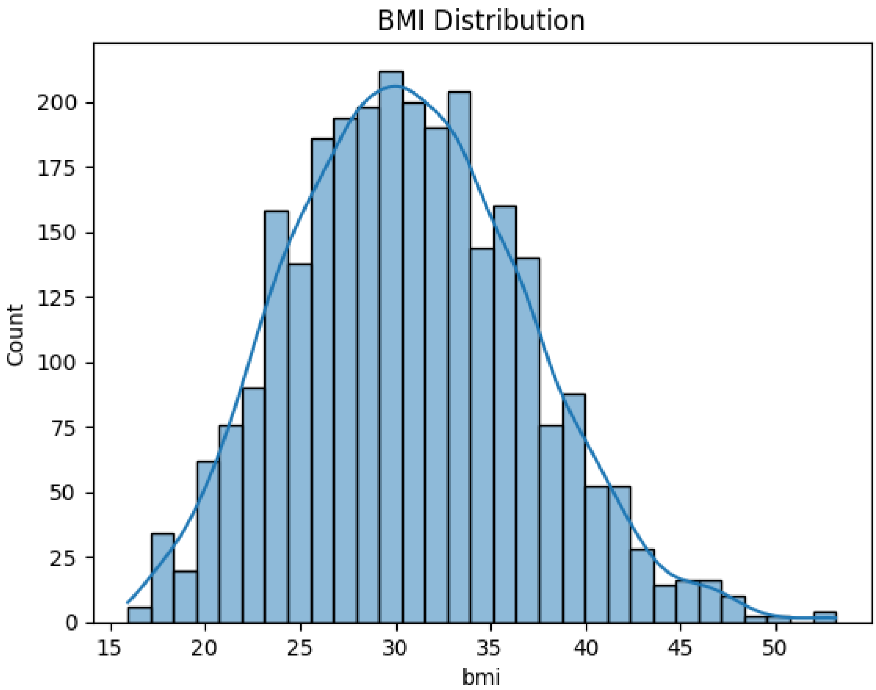

Figure 5: BMI Distribution.

Children Distribution: This attribute represents number of dependents covered by the insurance.

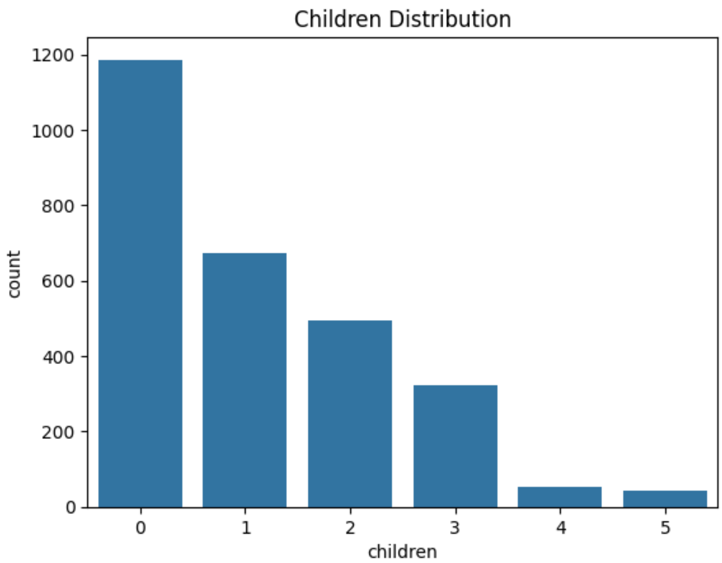

Figure 6: Children Distribution.

Smoker Distribution: It is binary categorical attribute with values yes and no.

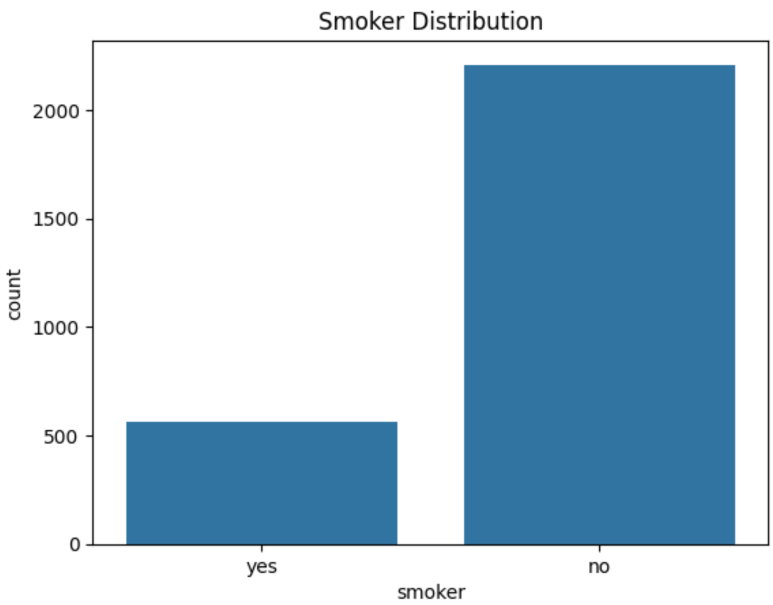

Figure 7: Smoker Distribution.

Region Distribution: This attribute holds 4 unique categorical values namely, northeast, northwest, southeast and southwest.

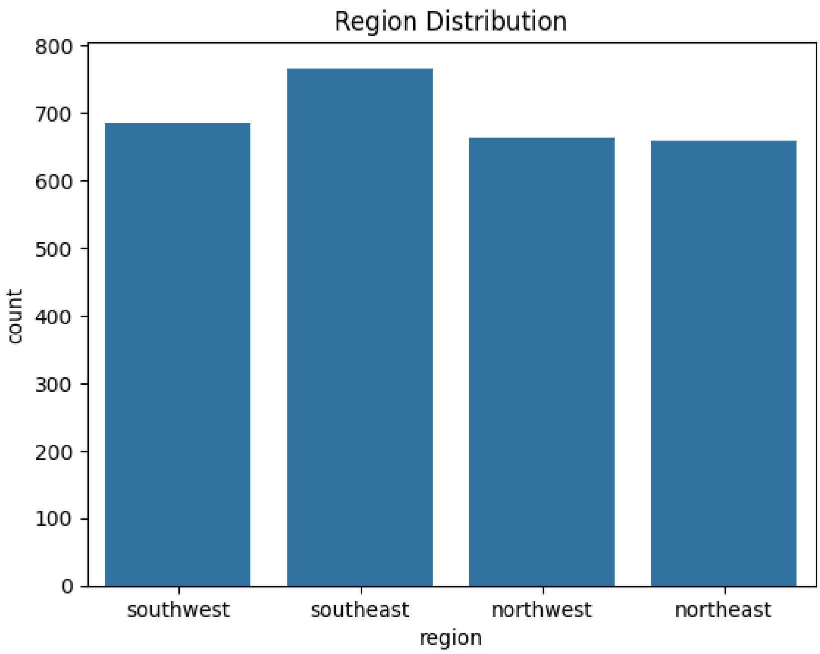

Figure 8: Region distribution.

4.3 Understanding Data:

HeatMap:

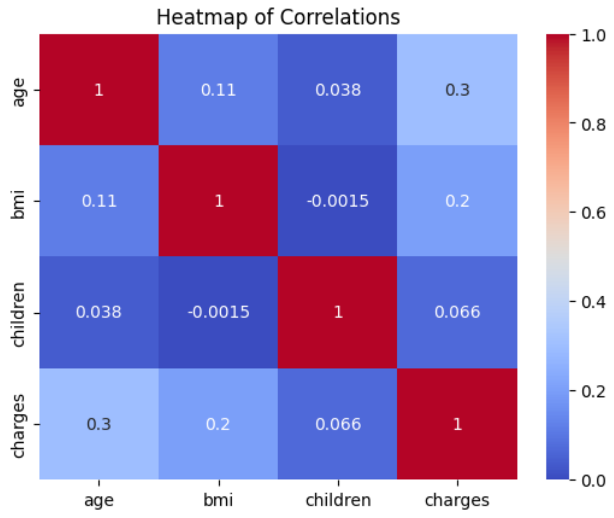

Figure 9: Heatmap of the variables.

Outliers:

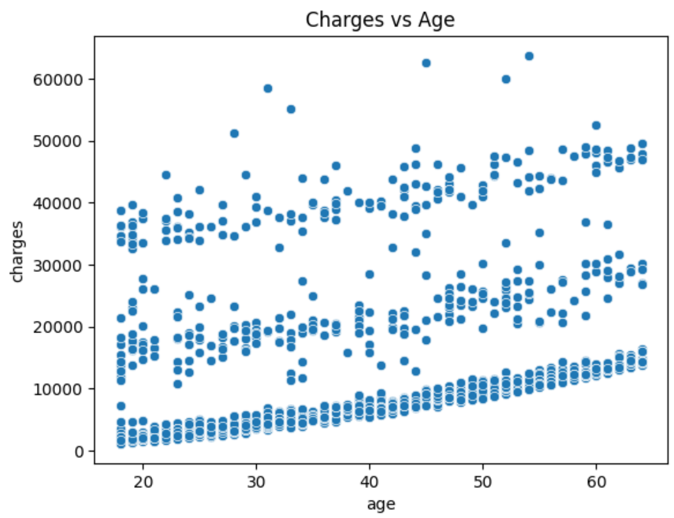

Figure 10: Scatter plot of outliers in Charges by Age.

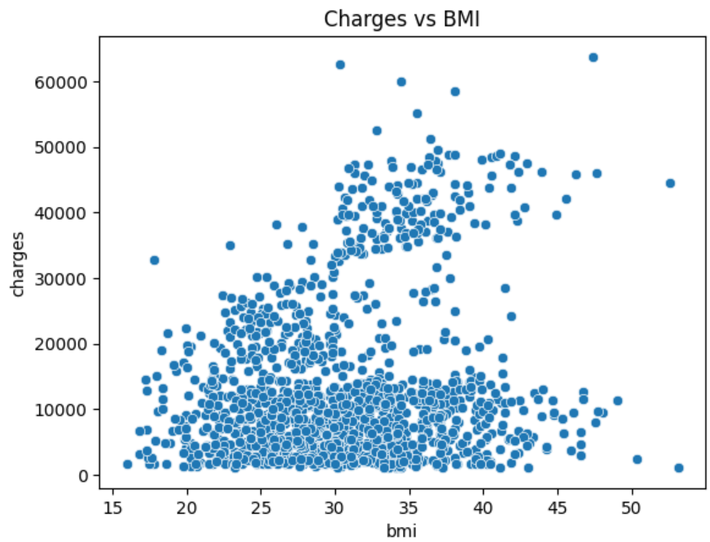

Figure 11: Scatter plot of outliers in Charges by BMI.

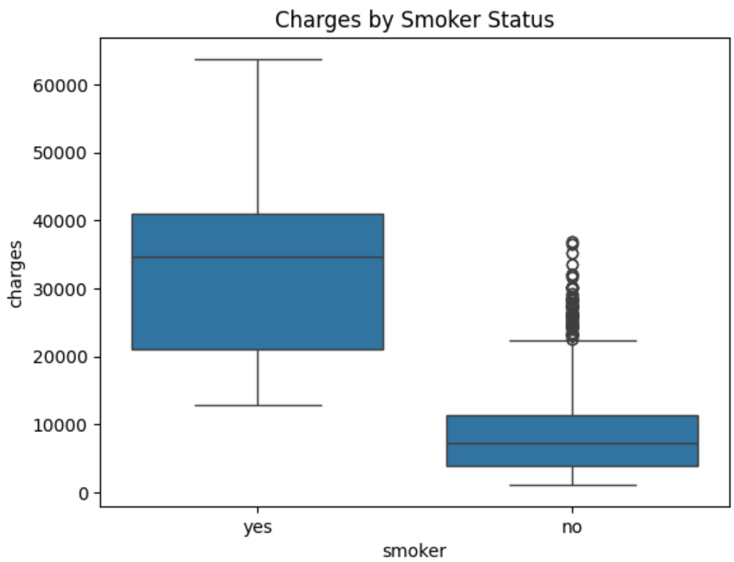

Figure 12: Box plot of Charges by Smoker status.

5 Model Implementation:

The models used for this research are Linear Regression, Random Forest, Ridge Regression and Gradient boosting.

Code-

```python
from sklearn.model_selection import GridSearchCV···

# Define models
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor()
}

# Define parameter grids for hyperparameter tuning
param_grids = {
    'Linear Regression': {},
    'Ridge Regression': {
        'model__alpha': [0.1, 1.0, 10.0]
    },
    'Random Forest': {
        'model__n_estimators': [100, 200],
        'model__max_depth': [None, 10, 20]
    },
    'Gradient Boosting': {
        'model__n_estimators': [100, 200],
        'model__learning_rate': [0.01, 0.1, 0.2]
    }
}
```

```python
# Train and tune models
results = {}
for name, model in models.items():
    print(f"Training {name}...")

    # Create a pipeline
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('model', model)])

    # Define GridSearchCV
    grid_search = GridSearchCV(pipeline, param_grids[name], cv=5, scoring='r2')
    grid_search.fit(X_train, y_train)

    # Get the best model and its performance
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_test)

    results[name] = {
        'Best Parameters': grid_search.best_params_,
        'R²': r2_score(y_test, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_test, y_pred)),
        'MAE': mean_absolute_error(y_test, y_pred),
        'MSE': mean_squared_error(y_test, y_pred)
    }
# Convert results to DataFrame for heatmap
results_df = pd.DataFrame(results).T
print(results_df)
```

Figure 13 : Model Training and Hyperparameter Tuning code.

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import learning_curve

# Assuming results_df is already created and populated with metrics
results_df = results_df.apply(pd.to_numeric, errors='coerce').fillna(0)

# Function definitions

# Residuals plot
def plot_residuals(model_name, model, X_test, y_test):
    y_pred = model.predict(X_test)
    residuals = y_test - y_pred
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_pred, y=residuals, alpha=0.7)
    plt.axhline(y=0, color='r', linestyle='--')
    plt.title(f'Residual Plot for {model_name}')
    plt.xlabel('Predicted Values')
    plt.ylabel('Residuals')
    plt.show()
```

```python
# Prediction vs Actual plot
def plot_predictions_vs_actual(model_name, model, X_test, y_test):
    y_pred = model.predict(X_test)
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred, alpha=0.7)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
    plt.title(f'Prediction vs Actual for {model_name}')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.show()

# Feature Importance plot
def plot_feature_importance(model_name, model, feature_names):
    if hasattr(model, 'feature_importances_'):
        importances = model.feature_importances_
        indices = np.argsort(importances)[::-1]
        plt.figure(figsize=(10, 6))
        sns.barplot(x=importances[indices], y=np.array(feature_names)[indices], palette='viridis')
        plt.title(f'Feature Importance for {model_name}')
        plt.xlabel('Importance')
        plt.ylabel('Features')
        plt.show()
```

```python
# Learning Curve plot
def plot_learning_curve(model_name, model, X, y):
    train_sizes, train_scores, test_scores = learning_curve(
        model, X, y, cv=5, scoring='r2', n_jobs=-1,
        train_sizes=np.linspace(0.1, 1.0, 10)
    )
    plt.figure(figsize=(10, 6))
    plt.plot(train_sizes, np.mean(train_scores, axis=1), 'o-', color='r', label='Training score')
    plt.plot(train_sizes, np.mean(test_scores, axis=1), 'o-', color='g', label='Cross-validation score')
    plt.title(f'Learning Curve for {model_name}')
    plt.xlabel('Training Size')
    plt.ylabel('Score')
    plt.legend()
    plt.show()
```

```python
# Plot for each model
for name, model in models.items():
    print(f"Plotting for {name}...")
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('model', model)])
    pipeline.fit(X_train, y_train)

    plot_residuals(name, pipeline, X_test, y_test)
    plot_predictions_vs_actual(name, pipeline, X_test, y_test)

    if isinstance(model, (RandomForestRegressor, GradientBoostingRegressor)):
        feature_names = preprocessor.transformers_[1][1].get_feature_names_out()
        feature_names = list(preprocessor.transformers_[0][1].get_feature_names_out()) + list(feature_names)
        plot_feature_importance(name, pipeline.named_steps['model'], feature_names)

    plot_learning_curve(name, pipeline, X, y)
```

Figure 14 : Coding for ploting the Model's Vistualization.

```
from sklearn.model_selection import cross_val_score

# Example for Random Forest
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('model', RandomForestRegressor())])
cv_scores = cross_val_score(pipeline, X, y, cv=5, scoring='r2')
print('Cross-Validation R² Scores:', cv_scores)
print('Average Cross-Validation R² Score:', np.mean(cv_scores))

Cross-Validation R² Scores: [0.97118273 0.98057437 0.97874154 0.97339556 0.98289248]
Average Cross-Validation R² Score: 0.977357335809576
```

Figure 15 : Model Validation code.

Linear Regression Model:

Visualization:

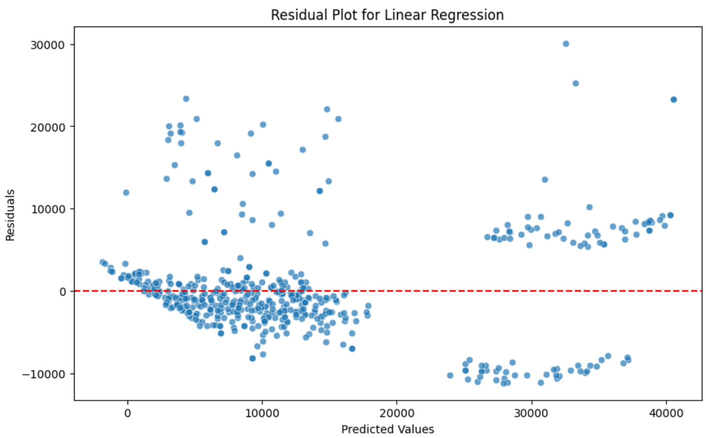

Figure 16: Residual plot for Linear Regression.

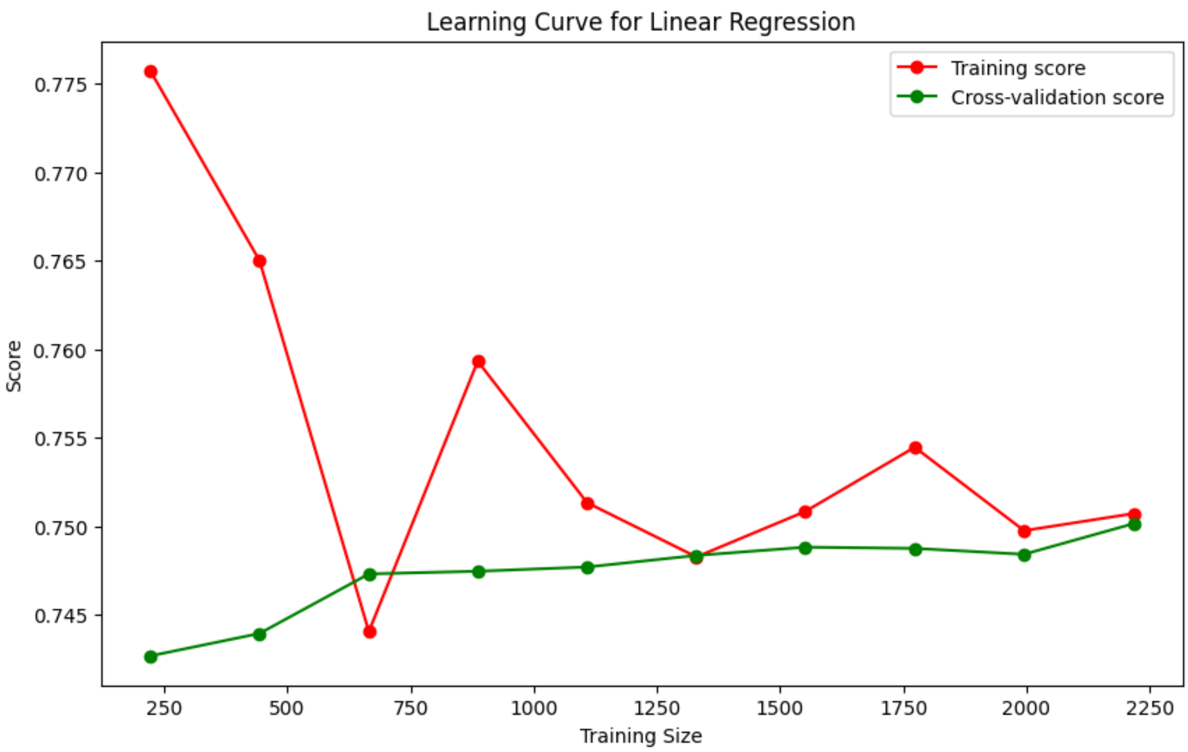

Figure 17: Training and Cross Validation score.

.

Random Forest Model:

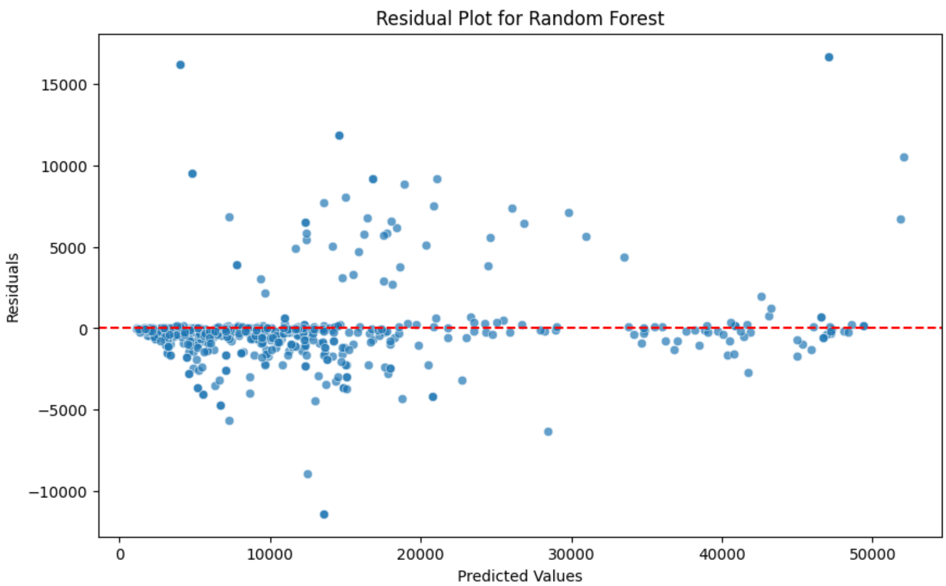

Figure 18: Residual Plot for Random Forest.

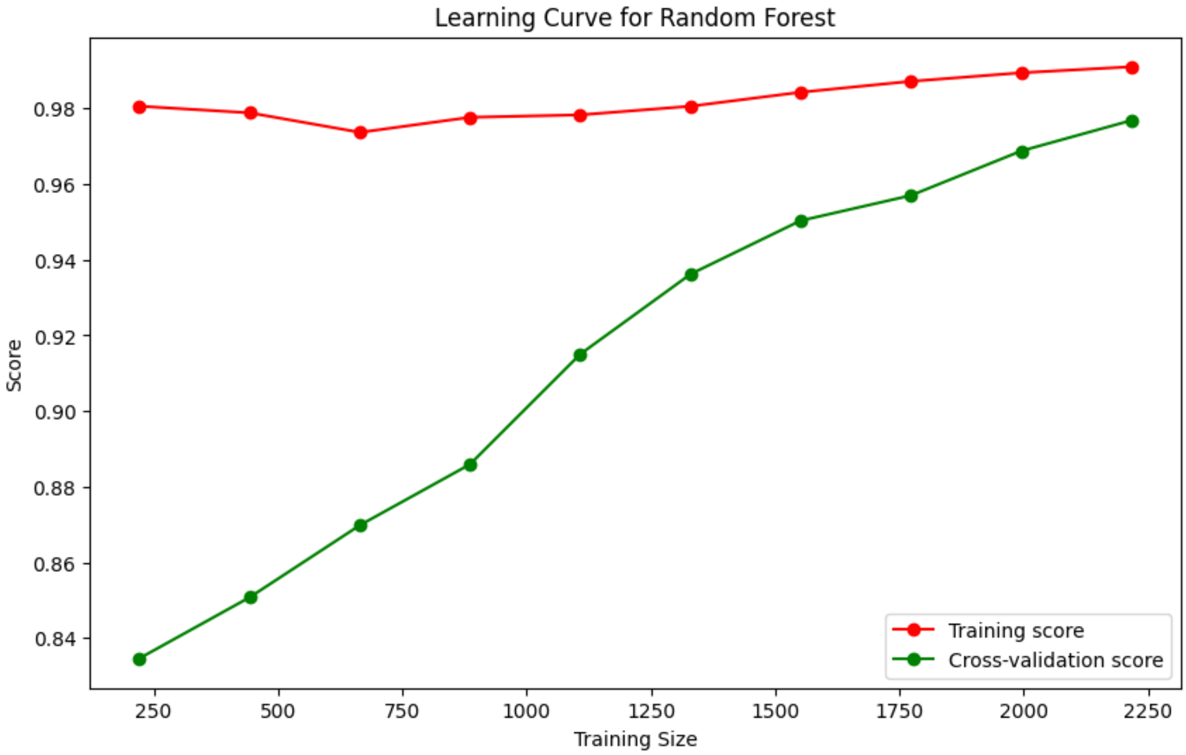

Figure 19: Training and Cross Validation Scores.

Gradient Booster Model:

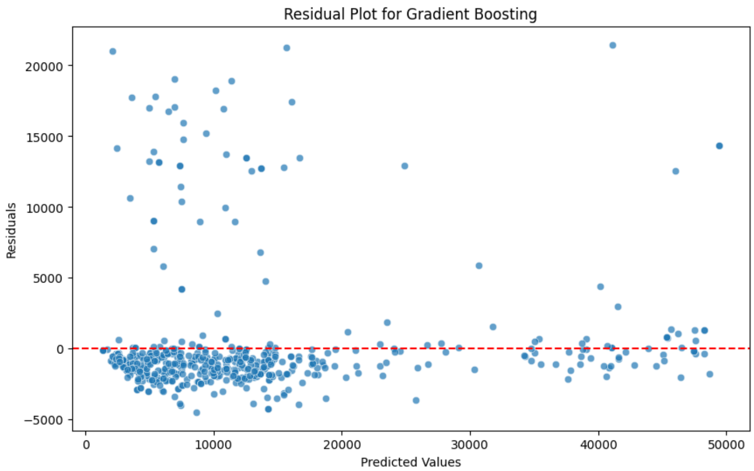

Figure 20: Residual plot for Gradient Boosting.

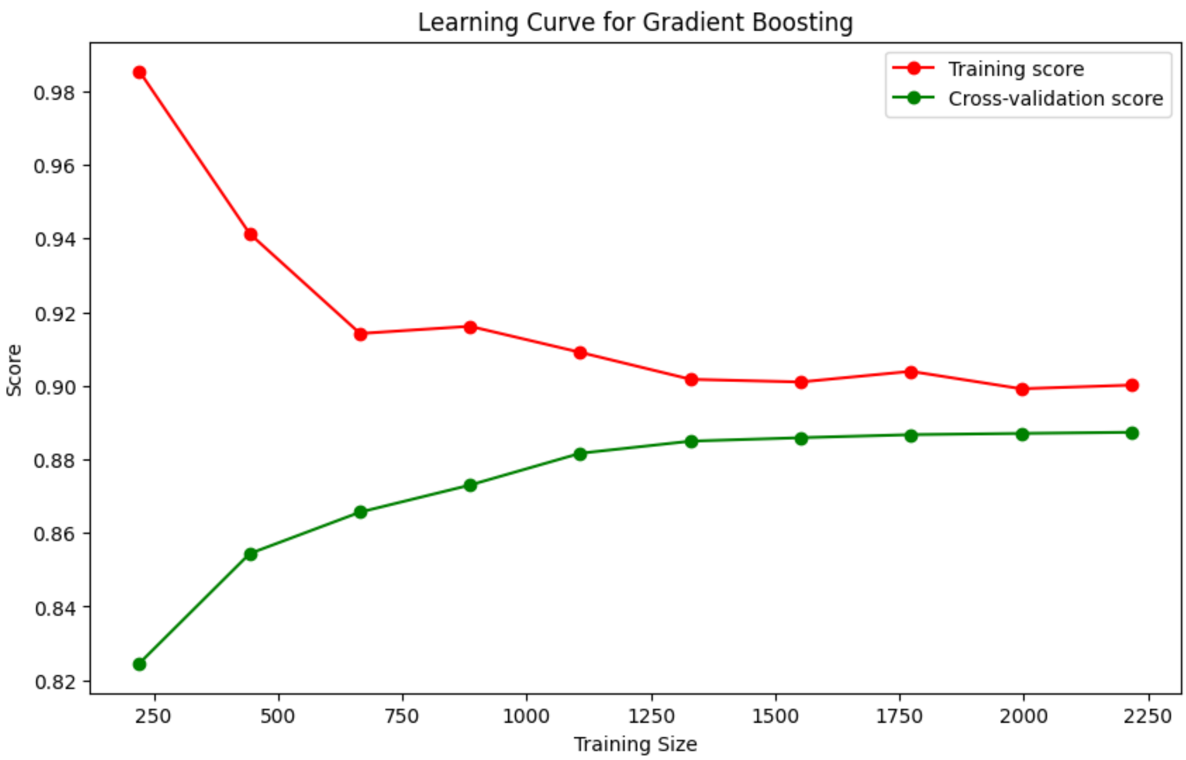

Figure 21: Training and Cross-Validation Score.

Ridge Regression Model:

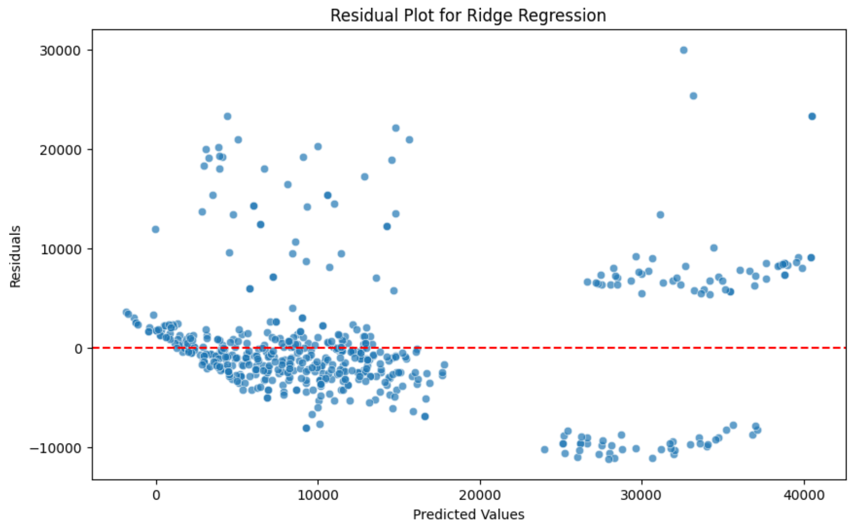

Figure 22: Residual Plot for Ridge Regression Model.

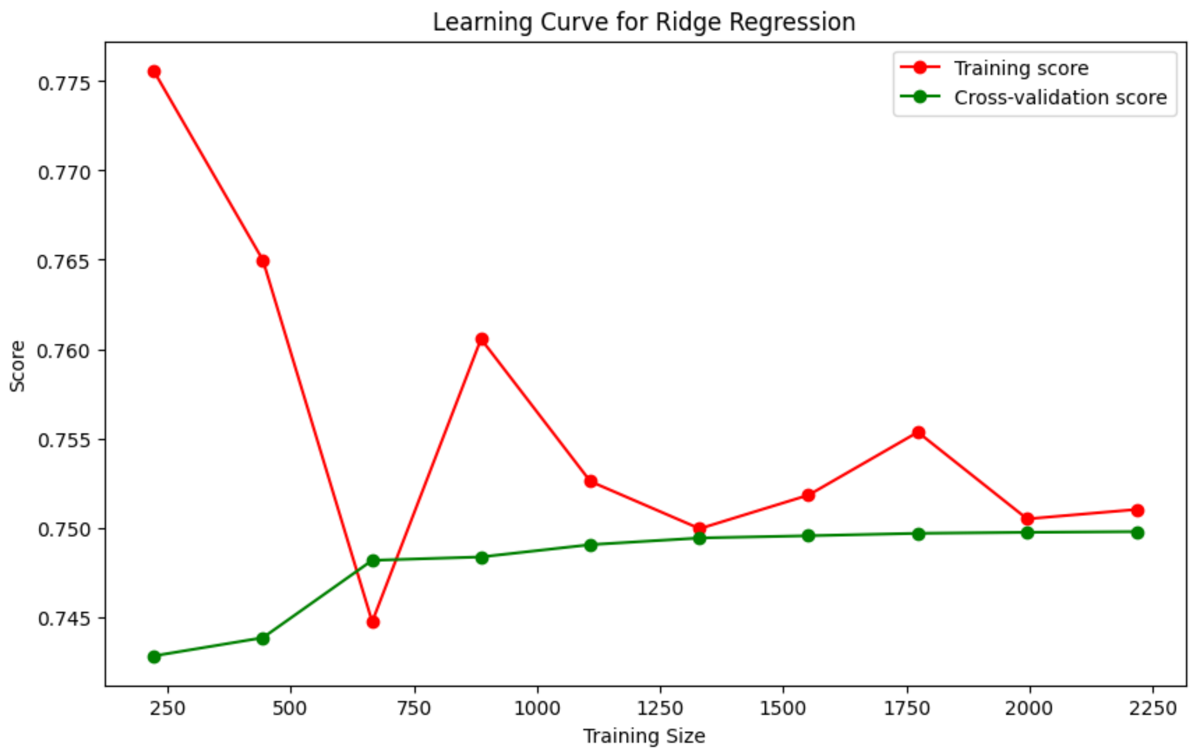

Figure 23: Training and Cross Validation Score.

8 Evaluation

R^2 square Visualization:

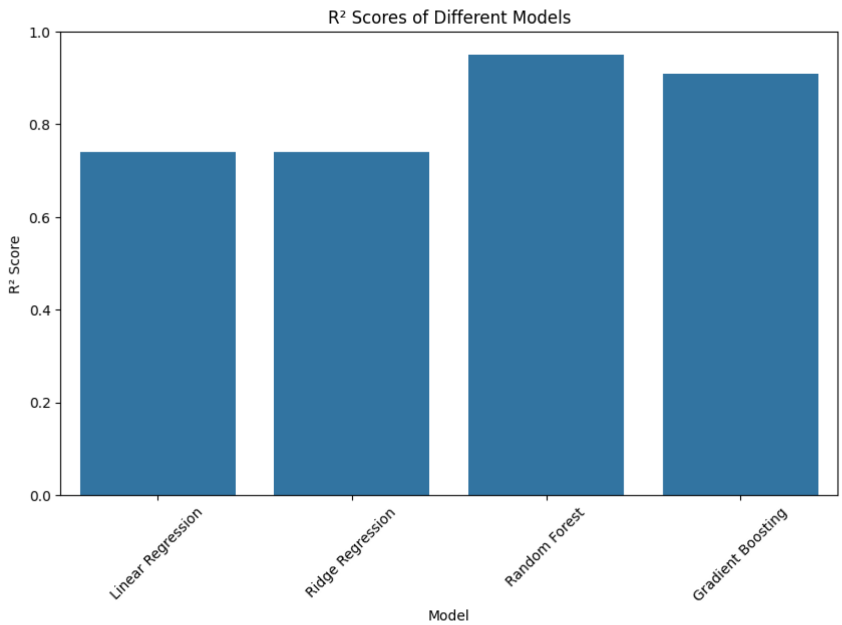

Figure 24: R^2 square Score comparison between models.

RMSE Visualization:

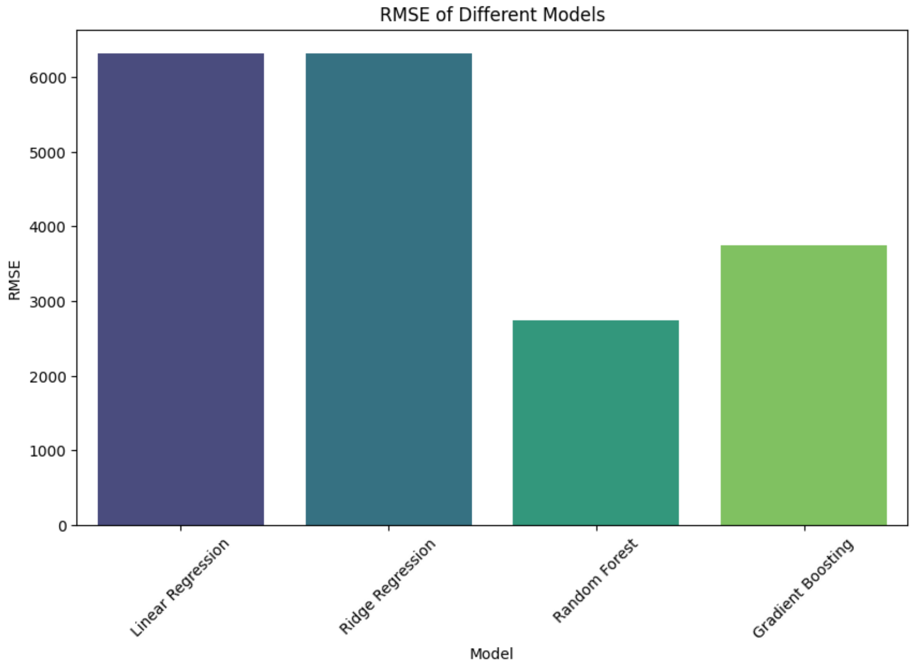

Figure 25: RMSE Score comparison between models.

MAE Visualization:

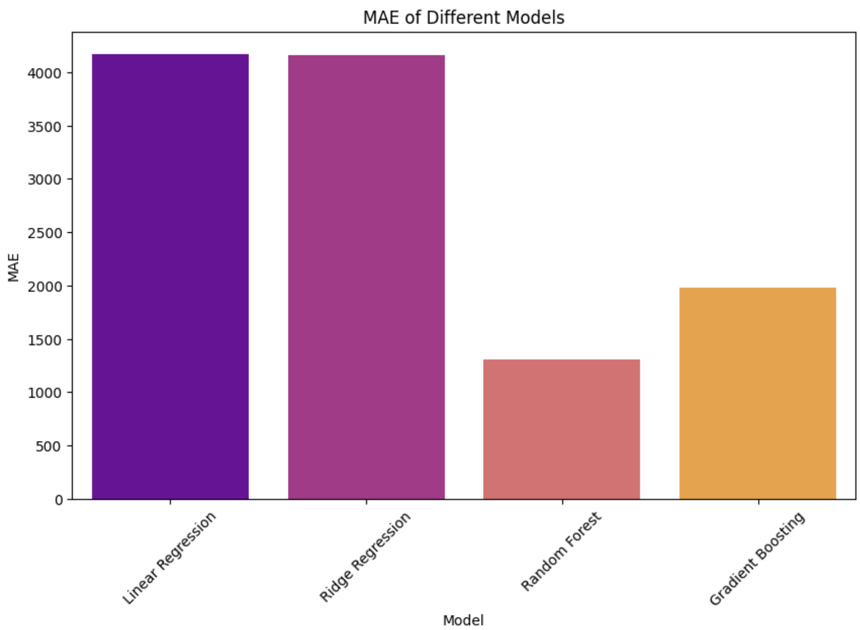

Figure 26: MAE Score comparison between models.

MSE Visualization:

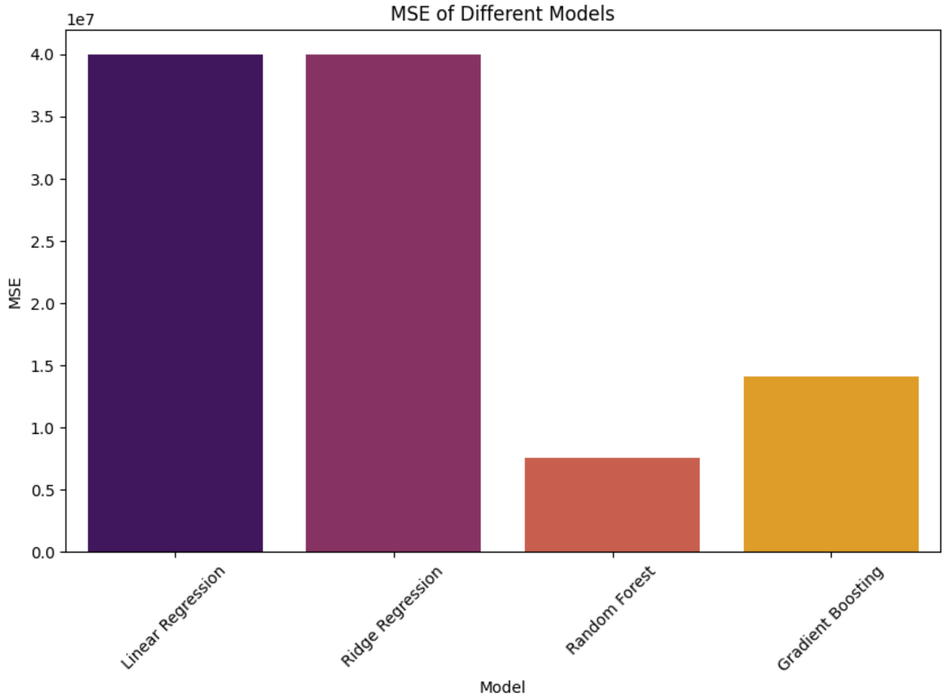

Figure 27: MSE Score comparison between models.

9 Predictions:

```python
# Predict using the best model (example with Random Forest)
best_model = RandomForestRegressor().fit(preprocessor.fit_transform(X), y)
y_pred = best_model.predict(preprocessor.transform(X_test))

# Compare predictions with actual values
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(comparison.head())
```

```
           Actual      Predicted
1106   8988.15875    9011.448047
1321  28101.33305   28165.113457
2313  12032.32600   12019.922340
2274   1682.59700    1818.717725
1432   3393.35635    4054.001614
```

Figure 28: Prediction code.

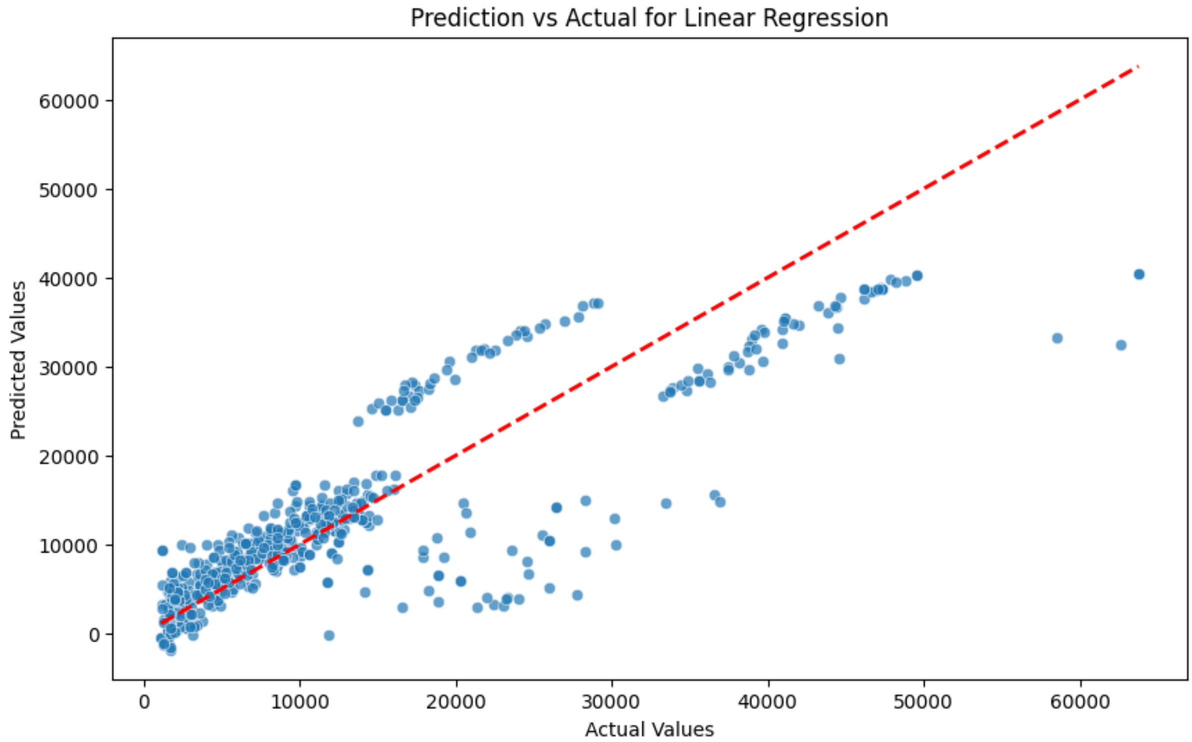

Figure 29: Plotting Prediction (blue dots) vs Actual values of Linear Regression.

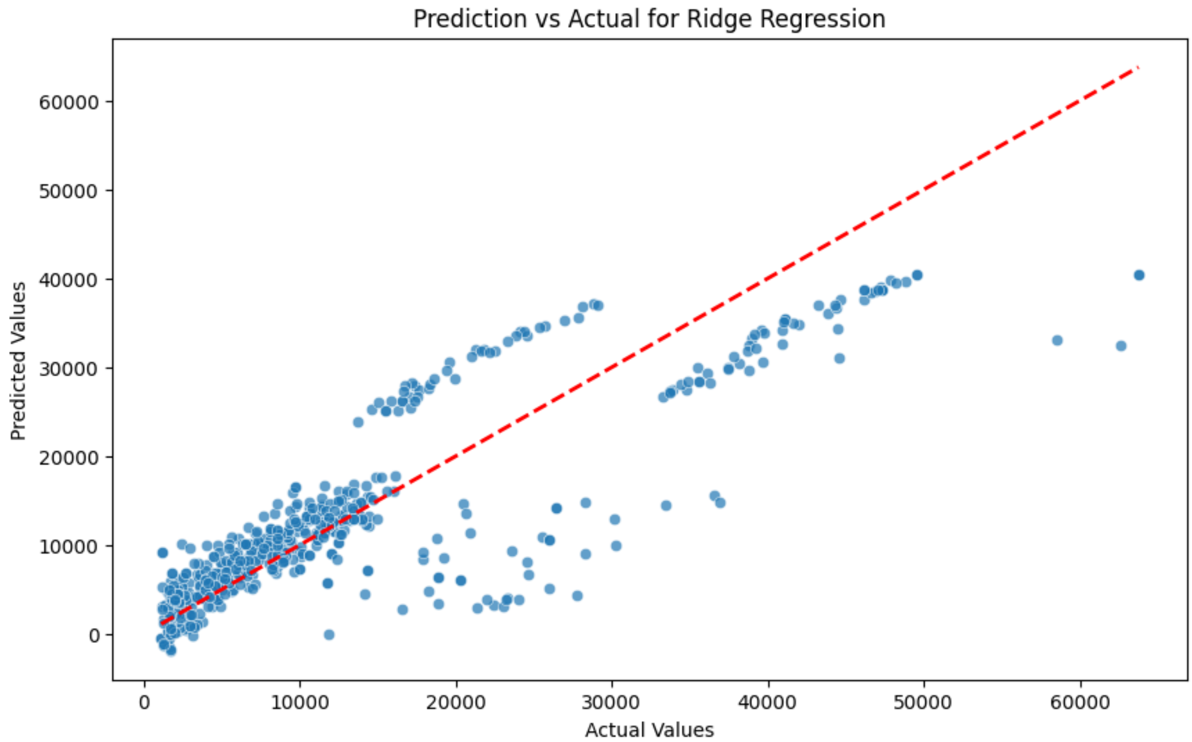

Figure 30: Plotting Prediction (blue dots) vs Actual values of Ridge Regression.

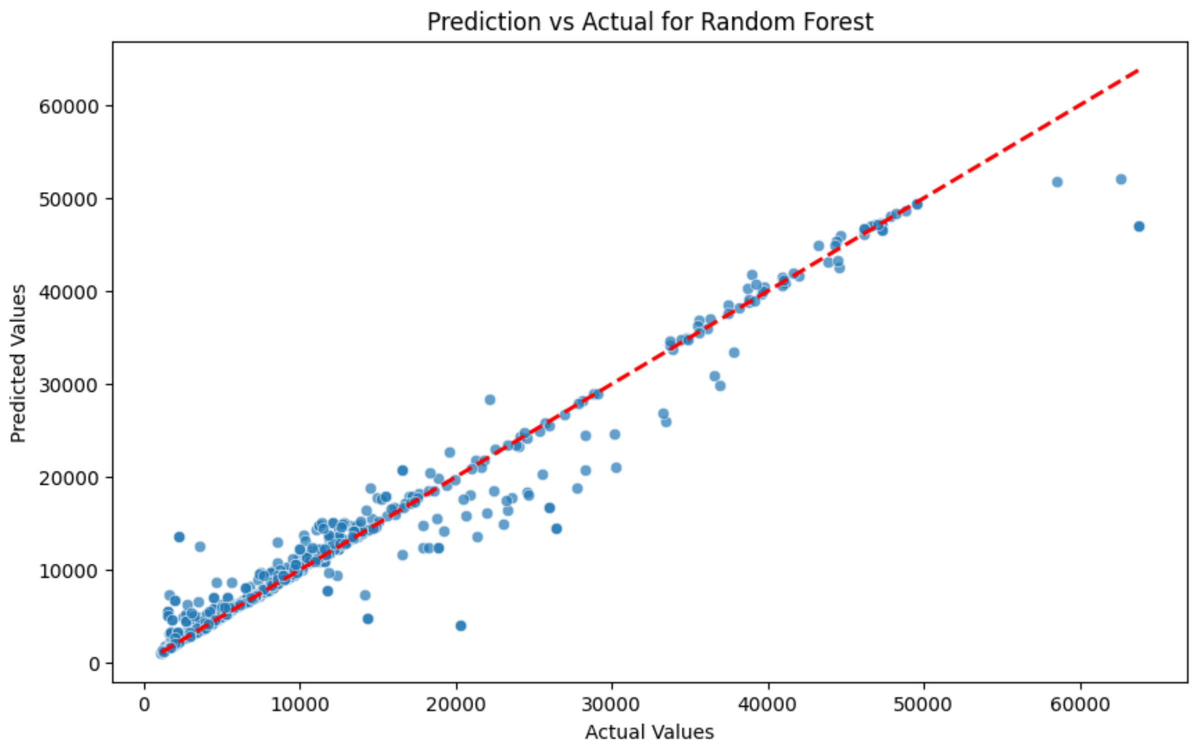

Figure 31: Plotting Prediction (blue dots) vs Actual values of Random Forest.

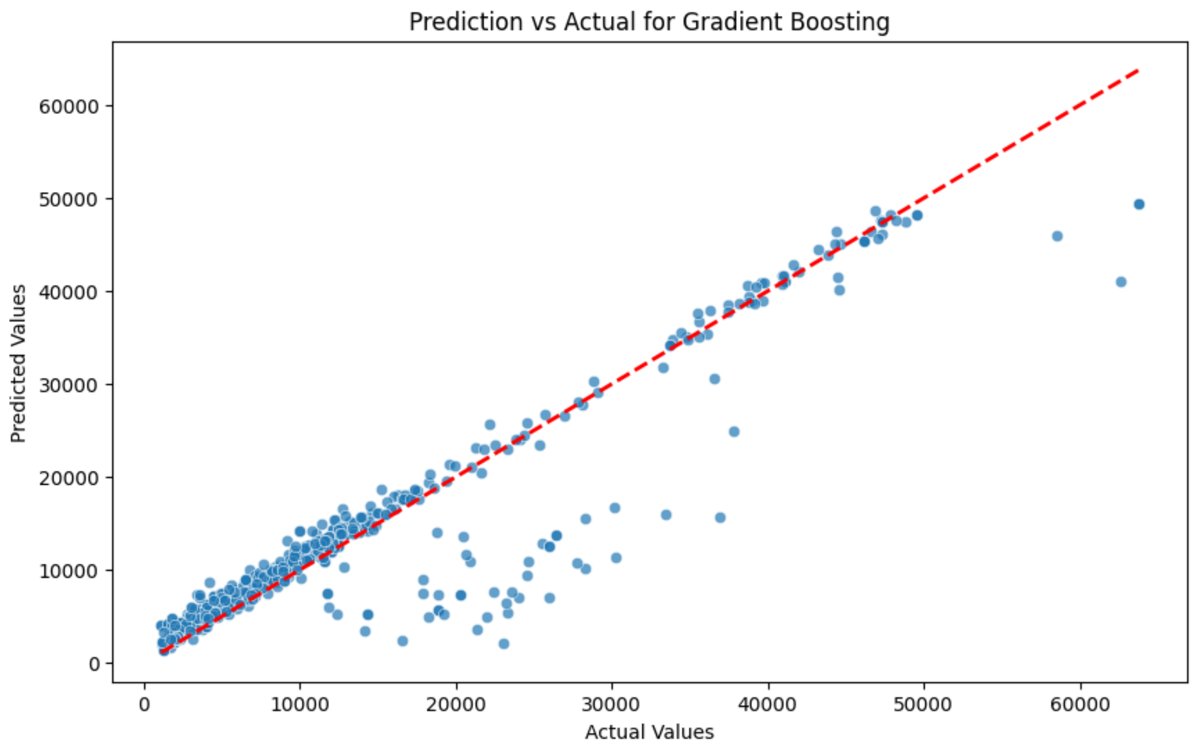

Figure 32: Plotting Prediction (blue dots) vs Actual values of Gradient Boosting.

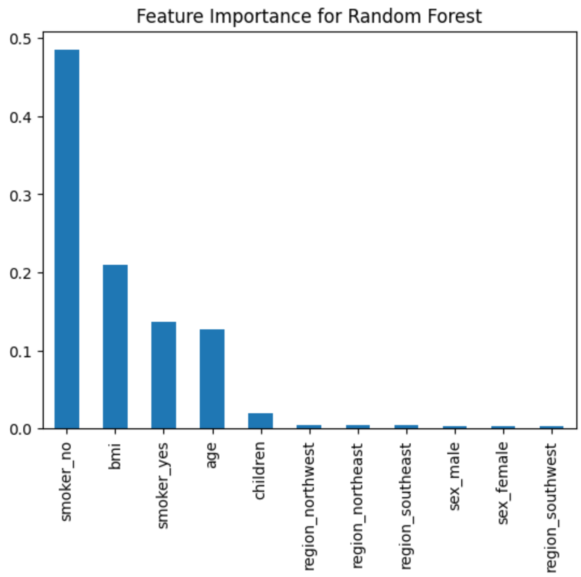

Figure 33: Feature Importance of the best model.

8.1 Interactive GUI App:

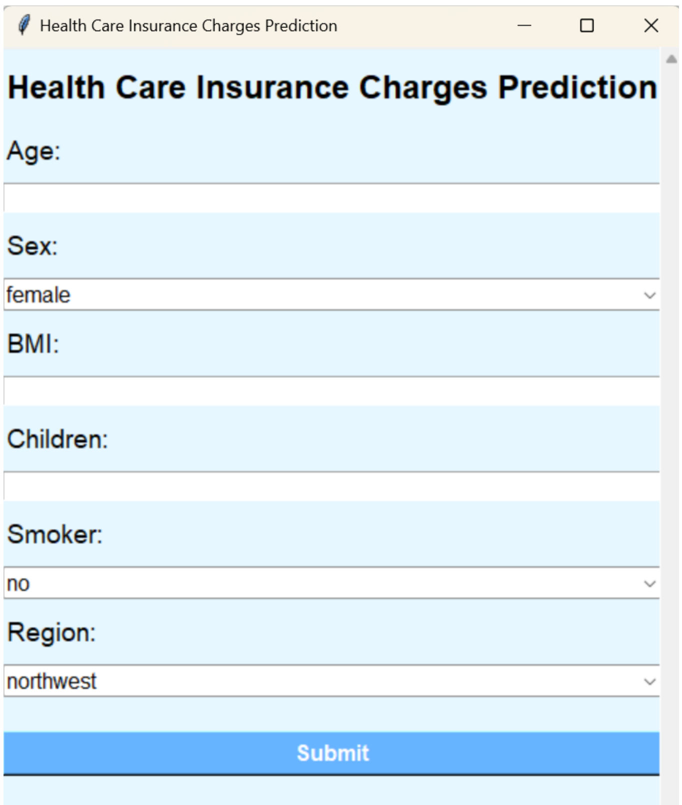

Figure 34: GUI.

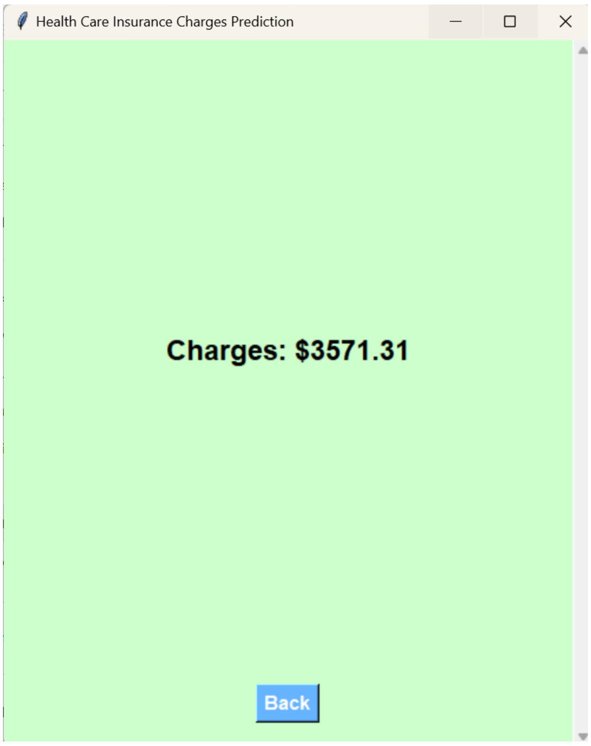

Figure 35: Prediction of Health Insurance Charges

References