Shaheed Zulfikar Ali Bhutto
Institute of Science & Technology

“PROJECT REPORT”

COURSE: ARTIFICIAL INTELLIGENCE

Group Members:

Roha Ali (2112124)

Hussam Wasti (2112113)

Respected Teachers:

Wali Muhammad Khubaib (Lab)

Sheikh Usama Khalid (Theory)

December/2024

THE DATASET:

```
df = pd.read_csv('house_price_mumbai.csv')
df
```

[2]

	BHK	project	Location	City	Total sqft	price_sqft	price
0	3 BHK Apartment	SHREE KRISHNA SangamChembur	Chembur	Mumbai	984	31,000	3.05 Cr
1	2 BHK Apartment	Ekdanta 24 KaratKurla	Kurla	Mumbai	598	23,913	1.42 Cr
2	2 BHK Apartment	Liberty Bay VueMalad West	Malad West	Mumbai	738	21,000	1.54 Cr
3	3 BHK Apartment	Thalia Vrindavan FloraRasayani	Rasayani	Mumbai	644	10,676	68.75 L
4	2 BHK Apartment	Mayfair The ViewVikhroli	Vikhroli	Mumbai	582	24,914	1.45 Cr
...	...	...	...	...	...	...	...
3975	2 BHK Apartment	Global Prestige Wing EVasai	Vasai	Mumbai	966	4,968	48 L
3976	1 BHK Apartment	Unicorn Unicorn Global ArenaNaigaon East	Naigaon East	Mumbai	500	5,200	26 L
3977	1 BHK Apartment	Navkar Navkar CityNaigaon East	Naigaon East	Mumbai	610	5,573	34 L
3978	1 BHK Apartment	Navkar City Phase I Part 3Naigaon East	Naigaon East	Mumbai	610	5,245	32 L
3979	1 BHK Apartment	Navkar City Phase I Part 1Naigaon East	Naigaon East	Mumbai	590	6,101	36 L

3980 rows × 7 columns

```
print(df.isnull().sum())
```

```
BHK              0
project        853
Location         0
City             0
Total sqft       0
price_sqft       0
price            0
dtype: int64
```

DATA CLEANING:

In [4]:
```
le_BHK = LabelEncoder()
le_project = LabelEncoder()
le_location = LabelEncoder()
```

In [5]:
```
df['BHK_n'] = le_BHK.fit_transform(df['BHK'])
df['project_n'] = le_project.fit_transform(df['project'])
df['location_n'] = le_location.fit_transform(df['Location'])
```

In [7]:
```python
def convert_price(value):
    value = value.replace(',', '')  # Remove commas
    if 'Cr' in value:
        return float(value.replace(' Cr', '')) * 10**7
    elif 'L' in value:
        return float(value.replace(' L', '')) * 10**5
#     elif 'Million' in value:
#         return float(value.replace(' Million', '')) * 10**6
    else:
        return float(value)

# Apply the conversion function to the price column
df['price'] = df['price'].apply(convert_price)
df['price_sqft'] = df['price_sqft'].apply(convert_price)
df['Total sqft'] = df['Total sqft'].astype(float)
df
```

Out[7]:

	BHK	project	Location	City	Total sqft	price_sqft	price	BHK_n	project_n	location_n
0	3 BHK Apartment	SHREE KRISHNA SangamChembur	Chembur	Mumbai	984.0	31000.0	30500000.0	10	732	28
1	2 BHK Apartment	Ekdanta 24 KaratKurla	Kurla	Mumbai	598.0	23913.0	14200000.0	5	138	86
2	2 BHK Apartment	Liberty Bay VueMalad West	Malad West	Mumbai	738.0	21000.0	15400000.0	5	295	94
3	3 BHK Apartment	Thalia Vrindavan FloraRasayani	Rasayani	Mumbai	644.0	10676.0	6875000.0	10	901	127
4	2 BHK Apartment	Mayfair The ViewVikhroli	Vikhroli	Mumbai	582.0	24914.0	14500000.0	5	365	161
...	...	...	...	...	...	...	...	...	...	...
3975	2 BHK Apartment	Global Prestige Wing EVasai	Vasai	Mumbai	966.0	4968.0	4800000.0	5	163	154
3976	1 BHK Apartment	Unicorn Unicorn Global ArenaNaigaon East	Naigaon East	Mumbai	500.0	5200.0	2600000.0	1	920	105
3977	1 BHK Apartment	Navkar Navkar CityNaigaon East	Naigaon East	Mumbai	610.0	5573.0	3400000.0	1	395	105
3978	1 BHK Apartment	Navkar City Phase I Part 3Naigaon East	Naigaon East	Mumbai	610.0	5245.0	3200000.0	1	392	105
3979	1 BHK Apartment	Navkar City Phase I Part 1Naigaon East	Naigaon East	Mumbai	590.0	6101.0	3600000.0	1	391	105

3980 rows × 10 columns

In [8]:
```python
df = df.drop(['project', 'BHK', 'Location', 'City'], axis='columns')
df
```

Out[8]:

	Total sqft	price_sqft	price	BHK_n	project_n	location_n
0	984.0	31000.0	30500000.0	10	732	28
1	598.0	23913.0	14200000.0	5	138	86
2	738.0	21000.0	15400000.0	5	295	94
3	644.0	10676.0	6875000.0	10	901	127
4	582.0	24914.0	14500000.0	5	365	161
...	...	...	...	...	...	...

In [9]:
```
df.describe()
```

Out[9]:

	Total sqft	price_sqft	price	BHK_n	project_n	location_n
count	3980.000000	3980.000000	3.980000e+03	3980.000000	3980.000000	3980.000000
mean	895.417588	11938.309799	1.122412e+07	4.578141	583.916583	72.766583
std	688.831332	10154.348760	1.708743e+07	4.977120	316.768013	51.360368
min	127.000000	114.000000	1.250000e+05	0.000000	0.000000	0.000000
25%	590.000000	4818.500000	2.829500e+06	1.000000	302.000000	22.000000
50%	717.000000	9000.000000	6.400000e+06	4.000000	683.000000	71.000000
75%	1060.000000	16172.750000	1.330000e+07	5.000000	920.000000	105.000000
max	16000.000000	92592.000000	3.000000e+08	22.000000	967.000000	173.000000

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3980 entries, 0 to 3979
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Total sqft   3980 non-null   float64
 1   price_sqft   3980 non-null   float64
 2   price        3980 non-null   float64
 3   BHK_n        3980 non-null   int32  
 4   project_n    3980 non-null   int32  
 5   location_n   3980 non-null   int32  
dtypes: float64(3), int32(3)
memory usage: 140.1 KB
```

In [11]:
```
print(df.isnull().sum())
```

```
Total sqft    0
price_sqft    0
price         0
BHK_n         0
project_n     0
location_n    0
dtype: int64
```

INPUT, OUTPUT:

In [15]:
```
X = df.drop(['price'], axis='columns')
y = df['price']
```

In [16]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Algorithms Implemented:

Linear Regression

What is linear regression?

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation. For instance, suppose that you have data about your expenses and income for last year. Linear regression techniques analyze this data and determine that your expenses are half your income. They then calculate an unknown future expense by halving the known future income.

Why is linear regression important?

Linear regression models are relatively simple and provide an easy-to-interpret mathematical formula to generate predictions. Linear regression is an established statistical technique and applies easily to software and computing. Businesses use it to reliably and predictably convert raw data into business intelligence and actionable insights. Scientists in many fields, including biology and the behavioral, environmental, and social sciences, use linear regression to conduct preliminary data analysis and predict future trends. Many data science methods, such as machine learning and artificial intelligence, use linear regression to solve complex problems.

How does linear regression work?

At its core, a simple linear regression technique attempts to plot a line graph between two data variables, x and y. As the independent variable, x is plotted along the horizontal axis. Independent variables are also called explanatory variables or predictor variables. The dependent variable, y, is plotted on the vertical axis. You can also refer to y values as response variables or predicted variables.

Steps in linear regression

For this overview, consider the simplest form of the line graph equation between y and x; y=c*x+m, where c and m are constant for all possible values of x and y. So, for example, suppose that the input dataset for (x,y) was (1,5), (2,8), and (3,11). To identify the linear regression method, you would take the following steps:

1. Plot a straight line and measure the correlation between 1 and 5.
2. Keep changing the direction of the straight line for new values (2,8) and (3,11) until all values fit.

3. Identify the linear regression equation as y=3*x+2.
4. Extrapolate or predict that y is 14 when x is

```python
# Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)

# Evaluate Linear Regression
mse_lin = mean_squared_error(y_test, y_pred_lin)
r2_lin = r2_score(y_test, y_pred_lin)
print("Linear Regression MSE:", mse_lin)
print("Linear Regression R2:", r2_lin)
```

```
Linear Regression MSE: 61707663179553.2
Linear Regression R2: 0.7304763289411867
```

LASSO REGRESSION:

What is Lasso Regression?

Lasso Regression is an acronym in the term LASSO which stands for;

L: Least
A: Absolute
S: Shrinkage and
S: Selection
O: Operator

1. Understanding Lasso Regression

The full form of LASSO is the Least Absolute Shrinkage and Selection Operation. As the name suggests, LASSO uses the “shrinkage” technique in which coefficients are determined, which get shrunk towards the central point as the mean.

The LASSO regression in regularization is based on simple models that possess fewer parameters. We get a better interpretation of the models due to the shrinkage process. The shrinkage process also enables the identification of variables strongly associated with variables corresponding to the target.

2. What is Regularization?

Regularization resolves the overfitting problem, which affects the accuracy level of the model. Regularization is executed by the addition of the “penalty” term to the best-fit equation produced by the trained data. This technique can be effectively used to minimize the number of variables to maintain them in the model. Regularization can cater to various purposes, such as understanding simpler models that include sparse and group structure models.

3. Regularization Techniques

There are two important techniques in the regularization, which are Ridge Regression and Lasso Regression model. Both techniques are utilized to reduce the complexity of the model. The techniques are similar except in terms of penalty terms since the lasso regression uses absolute weights, whereas ridge regression uses the square of weights.

4. What is LASSO Regression?

Lasso regression is also called Penalized regression method. This method is usually used in machine learning for the selection of the subset of variables. It provides greater prediction accuracy as compared to other regression models. Lasso Regularization helps to increase model interpretation.

The less important features of a dataset are penalized by the lasso regression. The coefficients of this dataset are made zero leading to their elimination. The dataset with high dimensions and correlation is well suited for lasso regression.

```python
# Lasso Regression
lasso_reg = Lasso(alpha=0.1)  # You can tune the alpha parameter
lasso_reg.fit(X_train, y_train)
y_pred_lasso = lasso_reg.predict(X_test)

# Evaluate Lasso Regression
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
r2_lasso = r2_score(y_test, y_pred_lasso)
print("Lasso Regression MSE:", mse_lasso)
print("Lasso Regression R2:", r2_lasso)
```

[18] Python

```
Lasso Regression MSE: 61707663149937.55
Lasso Regression R2: 0.7304763290705405
```

RIDGE REGRESSION:

What is a Ridge?

Ridge regression is the method used for the analysis of multicollinearity in multiple regression data. It is most suitable when a data set contains a higher number of predictor variables than the number of observations. The second-best scenario is when multicollinearity is experienced in a set.

Variables Standardization in Ridge Regression

Variable's standardization is the initial procedure in ridge regression. Both the independent and dependent variables require standardization through subtraction of their averages and a division of the result with the standard deviations. It is common practice to annotate in a formula whether the variables therein are standardized or not.

Therefore, all ridge regression computations use standardized variables to avoid the notations on whether individual variables have been standardized. The coefficients can then be reverted to their original scales in the end.

```python
# Ridge Regression
ridge_reg = Ridge(alpha=1.0)  # You can tune the alpha parameter
ridge_reg.fit(X_train, y_train)
y_pred_ridge = ridge_reg.predict(X_test)

# Evaluate Ridge Regression
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)
print("Ridge Regression MSE:", mse_ridge)
print("Ridge Regression R2:", r2_ridge)
```

```
Ridge Regression MSE: 61707630900644.73
Ridge Regression R2: 0.7304764699274024
```

Decision Tree Regression:

Decision trees are a type of supervised machine learning algorithm that is used by the Train Using AutoML tool and classifies or regresses the data using true or false answers to certain questions. The resulting structure, when visualized, is in the form of a tree with different types of nodes—root, internal, and leaf. The root node is the starting place for the decision tree, which then branches to internal nodes and leaf nodes. The leaf nodes are the final classification categories or real values. Decision trees are easy to understand and are explainable.

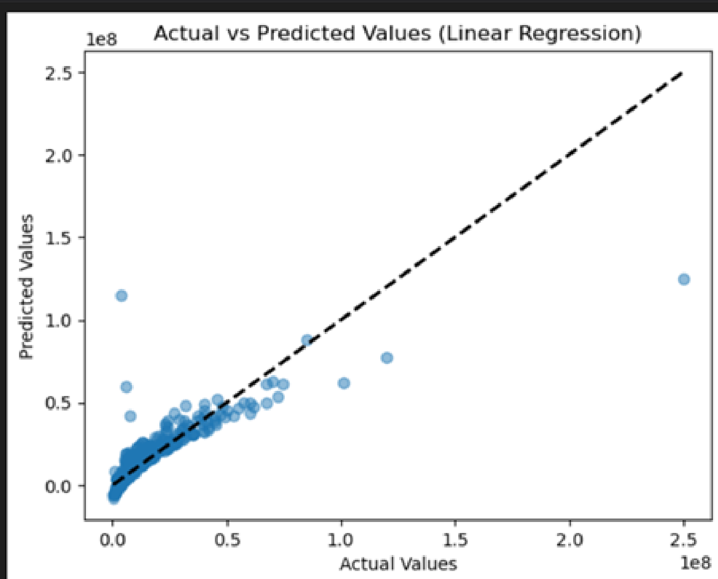
To construct a decision tree, start by specifying a feature that will become the root node. Typically, no single feature can perfectly predict the final classes; this is called impurity. Methods such as Gini, entropy, and information gain are used to measure this impurity and identify how well a feature classifies the given data. The feature with the least impurity is selected as the node at any level. To calculate Gini impurity for a feature with numerical values, first sort the data in ascending order and calculate the averages of the adjoining values. Then, calculate the Gini impurity at each selected average value by arranging the data points based on whether the feature values are less than or greater than the selected value and whether that selection correctly classifies the data. The Gini impurity is then calculated using the equation below, where K is the number of classification categories and p is the proportion of instances of those categories.

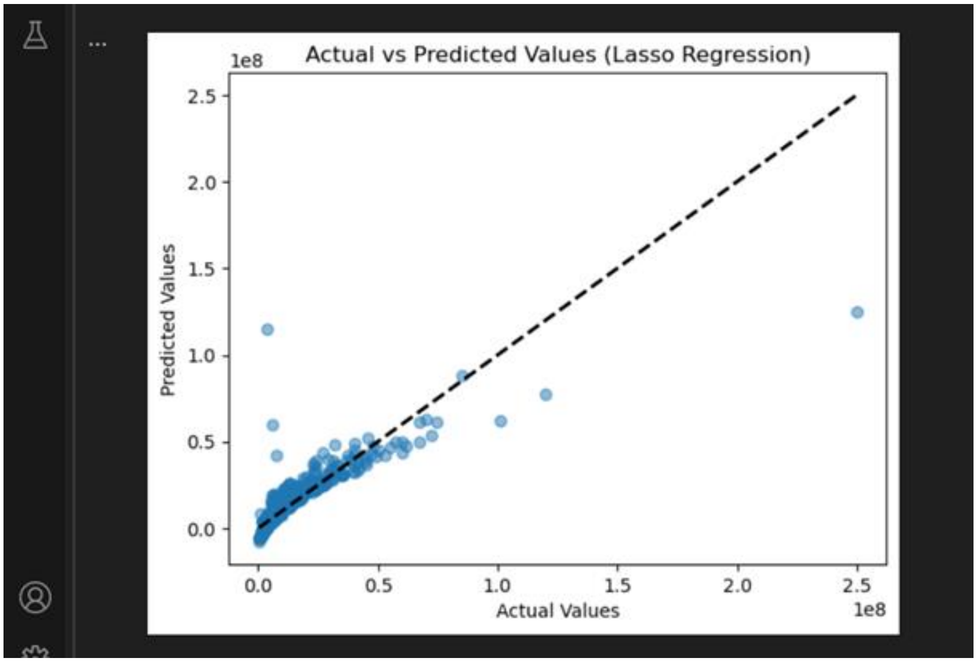
```python
# Decision Tree Regression
tree_reg = DecisionTreeRegressor(random_state=42)
tree_reg.fit(X_train, y_train)
y_pred_tree = tree_reg.predict(X_test)

# Evaluate Decision Tree Regression
mse_tree = mean_squared_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)
print("Decision Tree Regression MSE:", mse_tree)
print("Decision Tree Regression R2:", r2_tree)
```

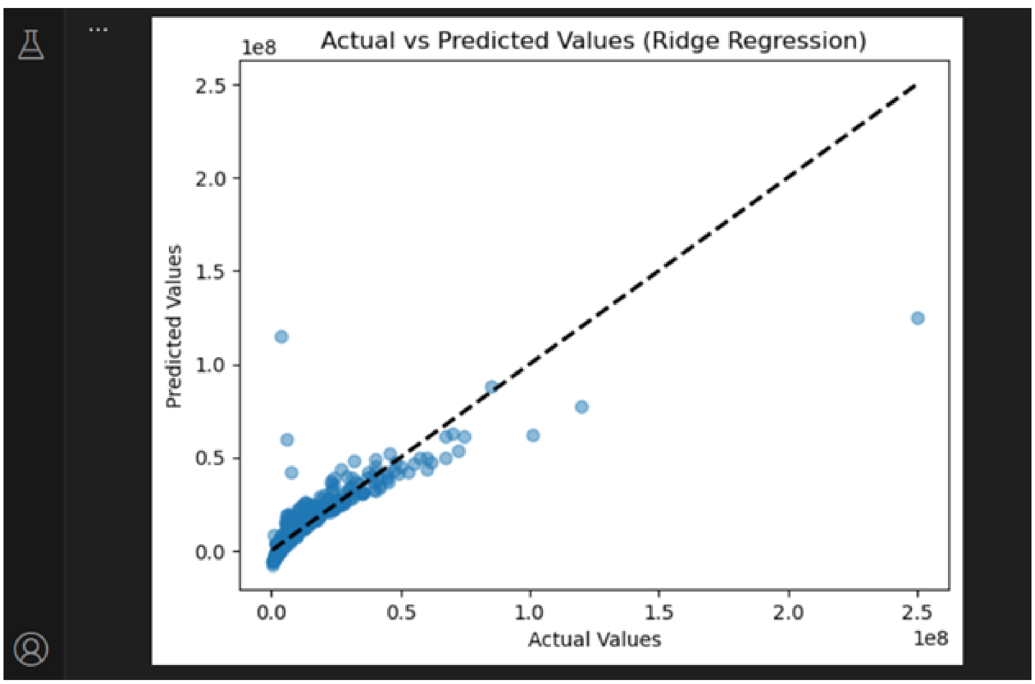
```
Decision Tree Regression MSE: 7234711711055.276
Decision Tree Regression R2: 0.9684005849688064
```

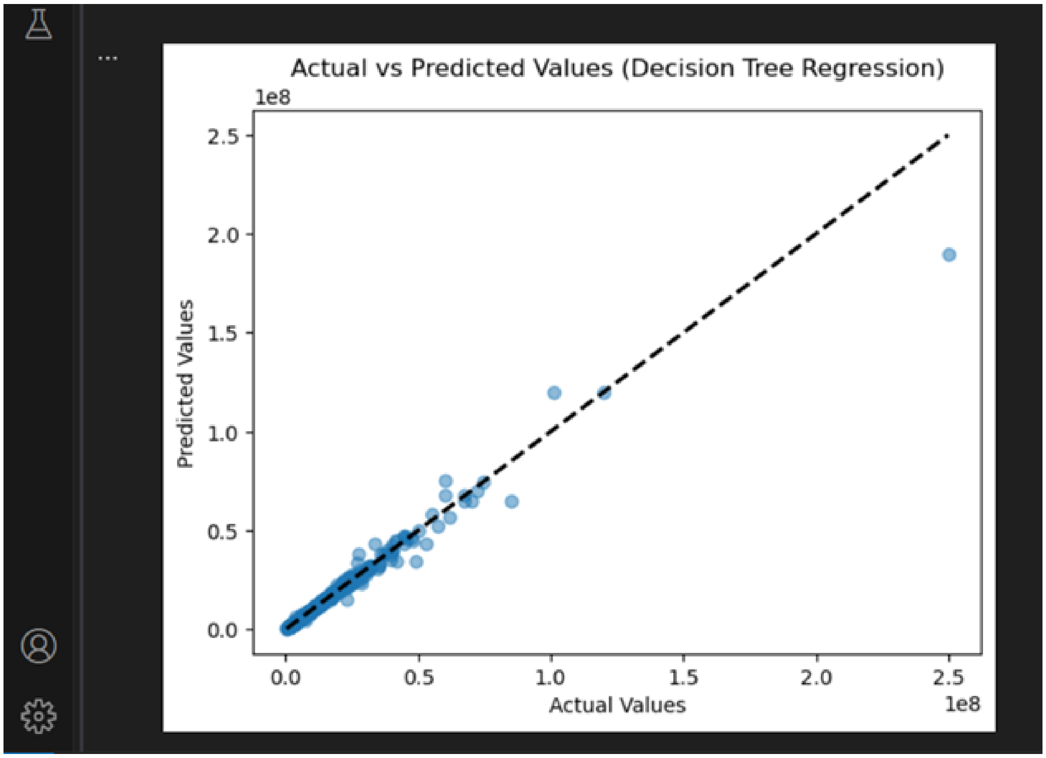
Graphs for comparative analysis:

```python
import matplotlib.pyplot as plt

# Function to plot actual vs predicted values
def plot_predictions(y_test, y_pred, model_name):
    plt.scatter(y_test, y_pred, alpha=0.5)
    plt.xlabel("Actual Values")
    plt.ylabel("Predicted Values")
    plt.title(f"Actual vs Predicted Values ({model_name})")
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
    plt.show()

# Plot predictions for each model
plot_predictions(y_test, lin_reg.predict(X_test), "Linear Regression")
plot_predictions(y_test, lasso_reg.predict(X_test), "Lasso Regression")
plot_predictions(y_test, ridge_reg.predict(X_test), "Ridge Regression")
plot_predictions(y_test, tree_reg.predict(X_test), "Decision Tree Regression")
```




Actual vs Predicted Values (Lasso Regression)
1e8
2.5
2.0
1.5
1.0
0.5
0.0
Predicted Values
0.0
0.5
1.0
1.5
2.0
2.5
Actual Values
1e8


Actual vs Predicted Values (Ridge Regression)
1e8
2.5
2.0
1.5
1.0
0.5
0.0
Predicted Values
0.0
0.5
1.0
1.5
2.0
2.5
Actual Values
1e8



Models Performance:

```python
# Summary of the models' performance
models = ['Linear Regression', 'Lasso Regression', 'Ridge Regression', 'Decision
mse_values = [mse_lin, mse_lasso, mse_ridge, mse_tree]
r2_values = [r2_lin, r2_lasso, r2_ridge, r2_tree]

summary_df = pd.DataFrame({
    'Model': models,
    'MSE': mse_values,
    'R2': r2_values
})

print(summary_df)
```

[24]

```
                      Model           MSE        R2
0         Linear Regression  6.170766e+13  0.730476
1          Lasso Regression  6.170766e+13  0.730476
2          Ridge Regression  6.170763e+13  0.730476
3  Decision Tree Regression  7.234712e+12  0.968401
```

CONCLUSION:

In conclusion, while each model has its strengths and weaknesses, Decision Tree Regression was the most effective for this specific dataset. It managed to capture the intricate relationships between features and the target variable better than linear models. This project highlights the importance of evaluating multiple models and understanding their underlying assumptions and characteristics to make informed decisions in predictive analytics.

Future work could involve tuning hyperparameters further, applying ensemble methods such as Random Forest or Gradient Boosting, or exploring advanced techniques like neural networks to potentially improve predictive performance.