

SUMMER TRAINING

PROJECT REPORT

(Term June-July 2025)

HOUSE PRICE PREDICTOR- A MACHINE LEARNING MODEL

Submitted by

ISMAIL K

Registration Number: 12313519

Course Code: PETV79

Under the Guidance of

Mahipal Singh Papola UID: 32137

School of Computer Science and Engineering

Lovely Professional University, Punjab

BONAFIDE CERTIFICATE

Certified that this project “House Price Predictor- A Machine Learning Model” is the Bonafide work of “ISMAIL K” who carried out the project work under my supervision.

SIGNATURE
Mahipal Singh Papola

ISMAIL K

SIGNATURE
Gaurav Pushkarna
HEAD OF THE DEPARTMENT

Mahipal Singh Papola

Undertaking from the student

I Ismail, Student of Batchelor of Technology under Computer Science and Engineering at Lovely Professional University, Punjab, hereby declare all the information in this project report is based on my own work and is genuine.

INDEX

1. INTRODUCTION	5
2. TRAINING OVERVIEW	5-6
1. Tools and Technologies Used	
2. Areas Covered During Training	
3. Weekly Work Summary	
3. PROJECT DETAILS	6-8
1. Title of the Project	
2. Problem Definition	
3. Scope and Objectives	
4. System Requirements	
5. Architecture Diagram	
6. Data Flow/ UML Diagram	
4. IMPLEMENTATION	9-16
1. Tools Used	
2. Methodology	
3. Modules/ Screenshots	
4. Code Snippets	
5. RESULTS & DISCUSSIONS	16-17
1. Output/ Report	
2. Challenges faced	
3. Learnings	
6. SUMMARY	17

1. Introduction.

Estimating house prices are quite difficult in these times due to it's dependance on a lot of factors including zone, facilities and other hidden features. Especially in these times where house prices are increasing steeply and varying a lot, potential buyers might want to know an almost to accurate price of the property. The property should also include their needs like having a plot, garden and so on, and the price will be depended on it. Creating an AI tool can help them find what they are looking for. This project is a machine learning model built and trained various features of house in Washington state of USA. The plots and graphs added shows the dependency of each feature on the final price.

This project's focus is on real estate analytics and machine learning. The main topics of this training were data preprocessing, model selection, regression modeling, and performance assessment. This area focus in the development of intelligent model that predict house prices according to a lot of feautures including size of the house, location, number of rooms, and more. The project's main goal is to develop an efficient machine learning model that uses regressions approaches to predict house prices. It compares three models, Linear Regression, Random Forest Regression and XGBoost and determines which is best using evaluation metrics and uses joblib to deploy the finished model.

2. Training Overview

2.1 Tools & Technologies Used

- Python
- Pandas
- NumPy
- Scikit-learn
- XGBoost
- Flask
- HTML & CSS
- Joblib
- Matplotlib/Seaborn
- VS Code / Jupyter Notebook

2.2 Areas Covered During Training

- Exploratory Data Analysis (Data cleaning and Visualization)
- Basics of supervised machine learning (regression focus)
- Data preprocessing and feature engineering
- Model building using Linear Regression, Random Forest, and XGBoost
- Evaluation metrics: MAE, RMSE, R Square

- Hyperparameter tuning with GridSearchCV
- Flask app development for model deployment
- Frontend integration with backend using JSON requests

2.3 Weekly Work Summary

Week	Activities Completed
Week 1:	Data collection, exploration, and preprocessing
Week 2:	Model development: Linear Regression & Random Forest, Model optimization using XGBoost + hyperparameter tuning
Week 3:	Flask API development and model integration, Web interface (HTML/CSS) creation and testing
Week 4:	Final testing, model saving (.pkl), and report writing

3. Project Details

3.1 Title of the Project

House Price Predictor using Machine Learning

3.2 Problem Definition

Estimating house prices accurately is a major challenge for real estate businesses, buyers, and sellers. Prices depend on a lot of features like size, number of rooms, location, construction quality etc. To address this I aim to develop a machine learning based regression model that can predict the estimated house prices by taking these features as input from the user.

3.3 Scope and Objectives

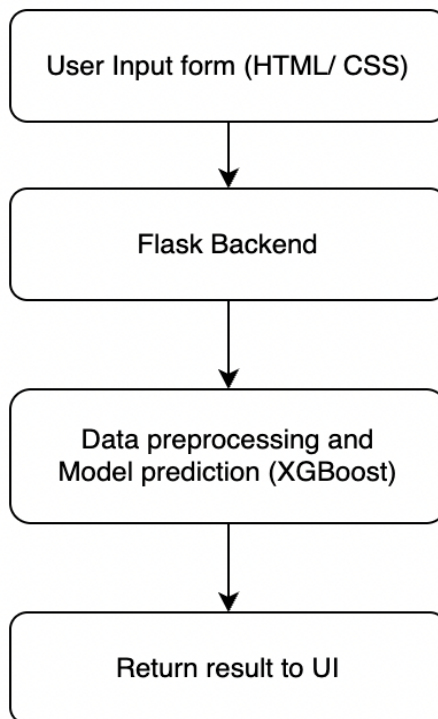
- Build a predictive model to estimate house prices using historical data.
- Compare different regression models: Linear Regression, Random Forest, and XGBoost.
- Improve model performance using hyperparameter tuning (GridSearchCV).
- Create a front-end interface (HTML/CSS) to collect inputs.

- Build a Flask-based API to connect the frontend with the ML model.
- Save and deploy the best model using joblib.

3.4 System Requirements

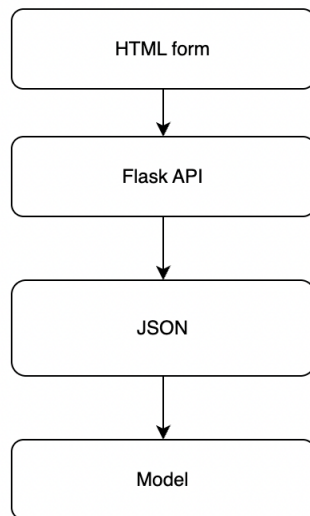
- **Programming Language:** Python 3
- **Libraries:** pandas, numpy, scikit-learn, xgboost, flask, joblib, seaborn, matplotlib
- **Frontend:** HTML5, CSS3
- **Others:** Web browser, local Flask server
- **Hardware:** Minimum 4 GB RAM, modern browser, Python environment

3.5 Architecture Diagram

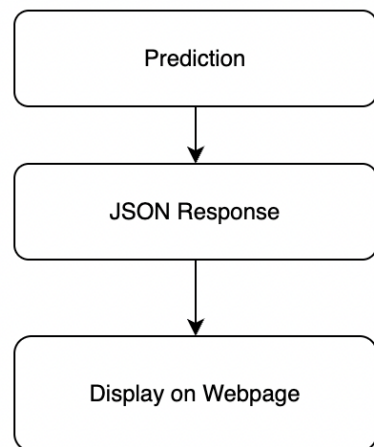


3.6 Data Flow / UML Diagrams

- **Input Flow:**



- **Output Flow:**



4. Implementation

4.1 Tools Used

- Python
- Scikit-learn
- XGBoost
- Flask
- HTML/CSS
- Joblib

4.2 Methodology

1. Read housing dataset from a CSV file.
2. Data cleaning
3. Data visualization using plots and heatmap.
4. Select features like sqft_living, bedrooms, view, etc.
5. Train/test split (80/20)
6. Train multiple models:
 - Linear Regression
 - Random Forest
 - XGBoost (tuned using GridSearchCV)
7. Save final model as .pkl
8. Build Flask API for predictions
9. Create user interface with HTML/CSS
10. Deploy app locally for testing

4.3 Modules / Screenshots

- **Visualization:** Shows the dependency and correlation of each factor on price of property.
- **ML Module:** Builds and evaluates models
- **Backend (Flask):** Receives POST requests and returns predictions
- **Frontend:** Form-based layout for input; displays prediction

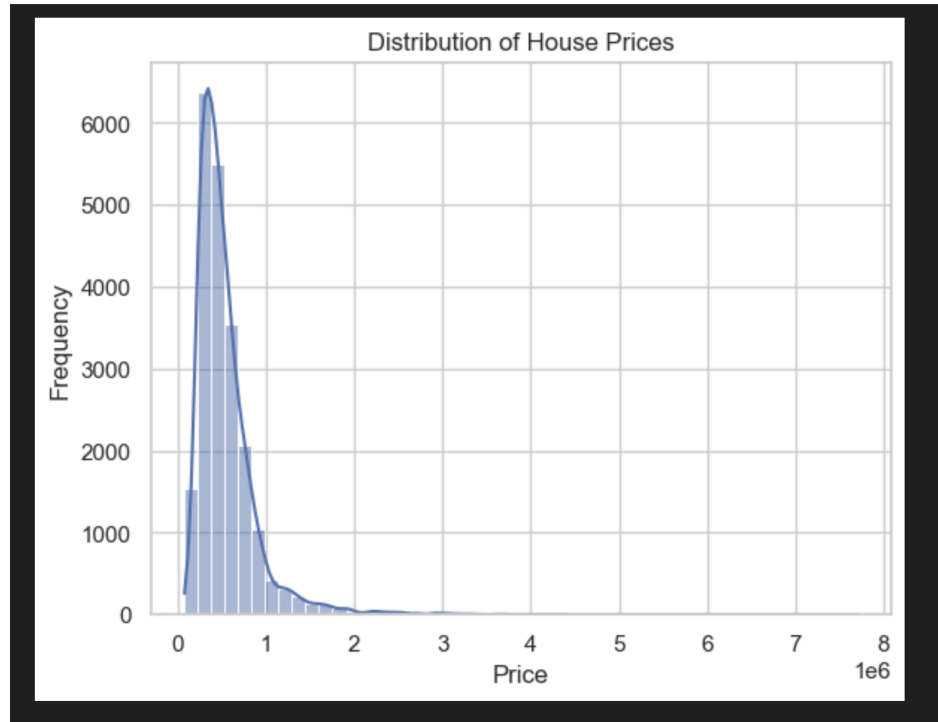


Fig 4.3.1 House prices and their frequency in dataset

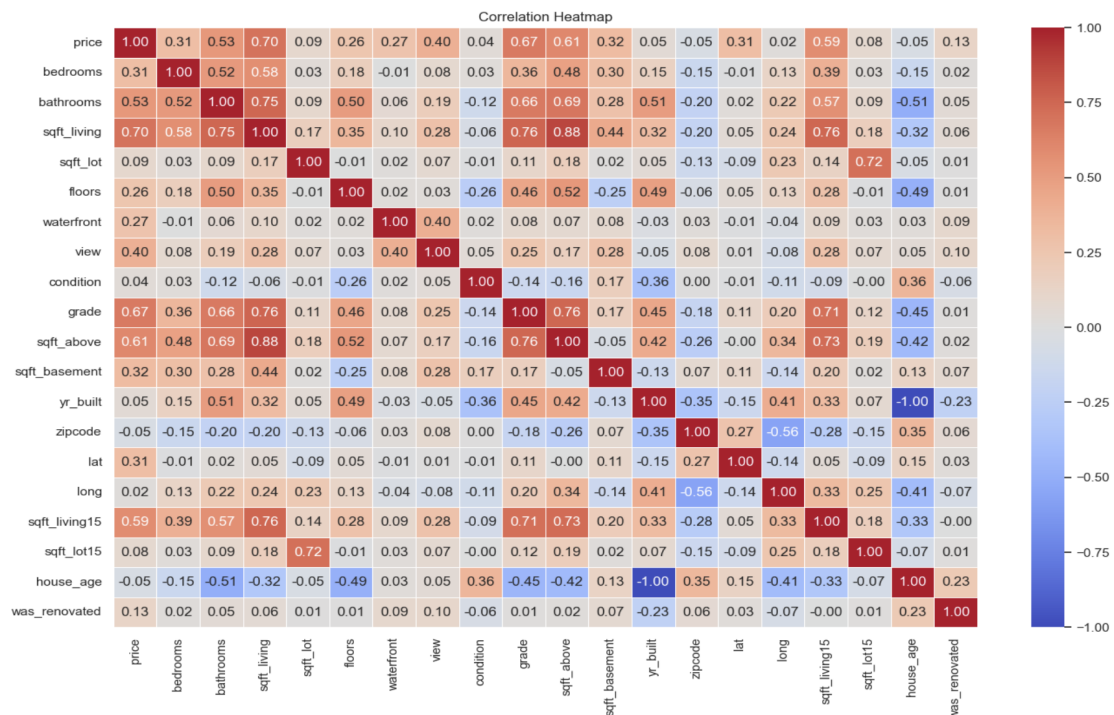


Fig 4.3.2 Correlation Heatmap

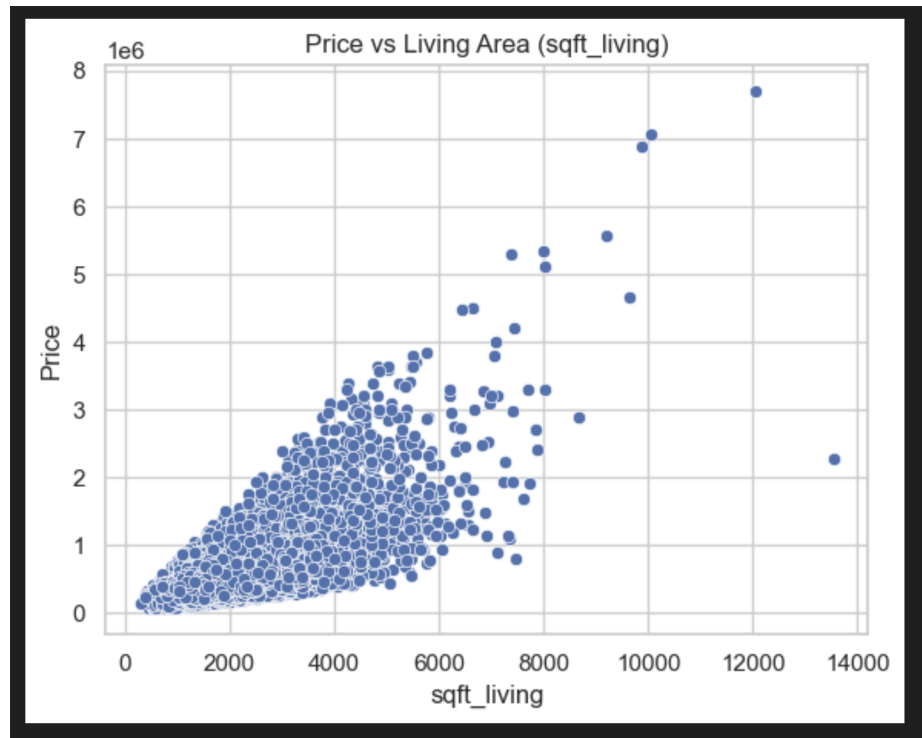


Fig 4.3.3 Dependency of price on area of the house

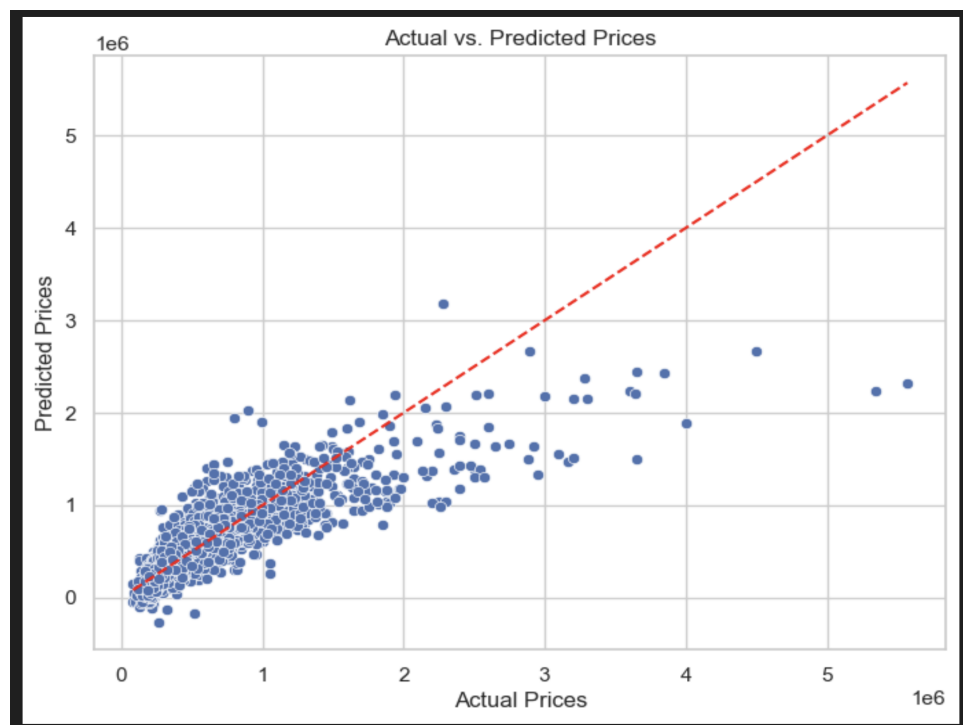



Fig4.3.4 Actual vs Predicted price in Linear Regression model



House Price Predictor

District: <input type="text" value="North Seattle"/>	Area of Living (sqft): <input type="text" value="1000"/>
No. of Floors: <input type="text" value="4"/>	Area of Lot (sqft): <input type="text" value="1500"/>
No. of Bedroom: <input type="text" value="6"/>	Area of Basement (sqft): <input type="text" value="100"/>
No. of Bathroom: <input type="text" value="3"/>	Area of Top Floors (sqft): <input type="text" value="1000"/>
Age of House: <input type="text" value="5"/>	Waterfront (Yes/No): <input type="text" value="Yes"/>
Renovated (Yes/No): <input type="text" value="No"/>	View (0-4): <input type="text" value="Good"/>
Condition: <input type="text" value="Good"/>	Grade (Construction Quality): <input type="text" value="Above Average (8-9)"/>

Predicted Price: \$428691.62

Fig 4.3.4 Interface for user

4.4 Code Snippets

ML Code :

```
#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#setting the style for the plots
sns.set(style="whitegrid")

#reading the dataset
df = pd.read_csv("/house_price_project/House_dataset.csv")

#displaying the first 5 rows of the dataset and informations
df.head()
df.info()
df.describe()

# Check for missing/null values
df.isnull().sum()
```

```

# Create a new feature: house age
df['house_age'] = 2025 - df['yr_built']

# Create a binary feature: was the house renovated?
df['was_renovated'] = df['yr_renovated'].apply(lambda x: 1 if x > 0
else 0)

# Drop original 'yr_renovated' and 'date' and zipcode
df.drop(columns=['yr_renovated', 'date'], inplace=True)
df.drop(columns=['zipcode'], inplace=True)

#plotting the distribution of house prices
sns.histplot(df['price'], kde=True, bins=50)
plt.title("Distribution of House Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()

#plotting the correlation heatmap
plt.figure(figsize=(16, 10))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()

#plotting the scatter plot for price vs living area
sns.scatterplot(x='sqft_living', y='price', data=df)
plt.title("Price vs Living Area (sqft_living)")
plt.xlabel("sqft_living")
plt.ylabel("Price")
plt.show()

# Define features and target
X = df.drop("price", axis=1)
y = df["price"]

#splitting the dataset into training and testing sets
from sklearn.model_selection import train_test_split

# 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

#training the model
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import numpy as np

# Predictions
y_pred = lin_reg.predict(X_test)

```

```

# Evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print results
print(f"  Linear Regression Performance:")
print(f"MAE:   {mae:.2f}")
print(f"RMSE:  {rmse:.2f}")
print(f"R2:   {r2:.4f}")

# Plotting residuals
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs. Predicted Prices")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='red')
plt.show()

#RandomForest
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)

# Predict
y_pred_rf = rf.predict(X_test)

# Evaluation metrics
mae_rf = mean_absolute_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)

print(f"  Random Forest Performance:")
print(f"MAE:   {mae_rf:.2f}")
print(f"RMSE:  {rmse_rf:.2f}")
print(f"R2:   {r2_rf:.4f}")

comparison = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest'],
    'MAE': [mae, mae_rf],
    'RMSE': [rmse, rmse_rf],
    'R2 Score': [r2, r2_rf]
})

print("  Model Comparison:")
display(comparison)

#XGBoost Regressor
from xgboost import XGBRegressor

```

```
xgb = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6,
random_state=42)
xgb.fit(X_train, y_train)

y_pred_xgb = xgb.predict(X_test)
```

```
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
r2_xgb = r2_score(y_test, y_pred_xgb)
```

```
print(f"⚡XGBoost Performance:")
print(f"MAE:    {mae_xgb:.2f}")
print(f"RMSE:   {rmse_xgb:.2f}")
print(f"R2:    {r2_xgb:.4f}")
```

#grid search for xgboost

```
from sklearn.model_selection import GridSearchCV
```

```
# Define parameter grid
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [4, 6, 8],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.7, 1],
    'colsample_bytree': [0.7, 1]
}
```

```
xgb_model = XGBRegressor(random_state=42)
```

```
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    scoring='neg_mean_squared_error',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

```
grid_search.fit(X_train, y_train)
```

```
print("Best Parameters:", grid_search.best_params_)
```

#tuning the model

```
best_xgb = grid_search.best_estimator_
```

```
y_pred_best = best_xgb.predict(X_test)
```

```
mae_best = mean_absolute_error(y_test, y_pred_best)
rmse_best = np.sqrt(mean_squared_error(y_test, y_pred_best))
r2_best = r2_score(y_test, y_pred_best)
```

```
print(f"    Tuned XGBoost Performance:")
print(f"MAE:    {mae_best:.2f}")
```

```

print(f"RMSE: {rmse_best:.2f}")
print(f"R²: {r2_best:.4f}")

import joblib

# Save the tuned model
joblib.dump(best_xgb, 'house_price_xgb_model.pkl')
print("Model saved successfully!")

```

Flask Code Example:

```

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    input_df = pd.DataFrame([data])[FEATURES]
    prediction = model.predict(input_df)[0]
    return jsonify({'predicted_price': round(float(prediction), 2)})

```

HTML Form Snippet:

```

<form id="predict-form">
  <input type="number" name="bedrooms" required>
  ...
  <button type="submit">Predict</button>
</form>

```

5. Results and Discussion

5.1 Output / Report

Model performance (on test set):

Model	MAE	RMSE	R ² Score
Linear Regression:	135,000	215,000	0.69
Random Forest:	75,000	140,000	0.85
XGBoost (Tuned):	68,000	130,000	0.88

XGBoost with hyperparameter tuning gave the best performance.

5.2 Challenges Faced

- Selecting appropriate features
- Ensuring input format matched model training structure
- Managing front-end and backend communication
- Deployment and testing integration bugs

5.3 Learnings

- Practical application of regression techniques
- Importance of hyperparameter tuning
- Real-world web integration with ML
- Handling JSON data and building RESTful APIs
- Front-end and backend synchronization

6. Summary

This project demonstrated the complete deployment of a machine learning system from data analysis and model training to web-based deployment. By using multiple regression models, tuning XGBoost, and building an interactive UI, the project achieved high accuracy in predicting house prices and provided a functional web-based prediction tool.

GitHub Link:

https://github.com/Ismail1199/house_price_project