DEBRE BIRHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

INDIVIDUAL ASSIGNMENT

Student Name: Etsub Zewdu

Student Id: DBU 1401215

Submited to:Tr. Derbew

Submission Date:10/10/2016 E.C

Machine Learning Individual Assignment Report

Project Title: Predicting House Prices Using Machine Learning

Course: Fundamentals of Machine Learning

---

1. Problem Definition

1.1 Objective

The objective of this project is to develop a machine learning model capable of accurately predicting house prices based on various property features. We will leverage regression analysis to build a model that can estimate housing prices based on key attributes, such as the number of bedrooms, square footage, location, and the year built. The model aims to assist stakeholders such as real estate agents, buyers, and investors in making informed decisions regarding property prices.

In this project, we will not only implement the model but also explore the underlying data to understand feature relationships, preprocess the data for accuracy, and deploy the model for real-time predictions.

1.2 Scope

The scope of this project includes the following aspects:

- **Exploratory Data Analysis (EDA):** Understanding the relationships between various property features and their impact on house prices.
- **Data Preprocessing:** Cleaning the dataset by handling missing values, encoding categorical features, and scaling numerical features.
- **Model Implementation and Training:** Developing a linear regression model to predict house prices.
- **Model Evaluation:** Using appropriate performance metrics like Mean Squared Error (MSE) to assess the model's prediction accuracy.

- **Deployment:** Deploying the trained model using FastAPI, allowing users to input property data and receive real-time price predictions.

The focus is on utilizing a linear regression model, though more advanced models may be considered in future iterations.

---

2. Data Source and Description

2.1 Source of Data

The dataset used in this project is self-sourced and mimics real-world housing data, containing key features commonly used in predicting house prices. This dataset serves as a proxy for real estate data and includes attributes such as the number of bedrooms, bathrooms, and square footage of the house.

2.2 License / Terms of Use

As the dataset is synthetic, it is free of any licensing restrictions. However, when working with real-world datasets, it's important to ensure compliance with the licensing terms provided by data sources such as Kaggle, government open data portals, or other platforms.

2.3 Data Structure

The dataset is structured in a CSV file, where each row represents a property and each column contains relevant features. Below is a summary of the features included:

Feature	Description	Type
bedrooms	Number of bedrooms	Integer
bathrooms	Number of bathrooms	Float
sqft_living	Total living area (square feet)	Float
sqft_lot	Total lot size (square feet)	Float
floors	Number of floors	Float
waterfront	Whether the property is by water (0 = No, 1 = Yes)	Integer
view	Rating of the view (0 = None, 4 = Excellent)	Integer
condition	Overall condition (1 = Poor, 5 = Excellent)	Integer
sqft_above	Square footage above ground level	Float
sqft_basement	Square footage of the basement	Float
yr_built	Year the house was built	Integer
yr_renovated	Year the house was last renovated	Integer
street	Street address of the property	String
city	City where the property is located	String

Feature	Description	Type
statezip	State and zip code	String
country	Country where the property is located	String
Target: price	Selling price of the house	Float

---

3. Exploratory Data Analysis (EDA)

3.1 Summary Statistics

We started by loading the dataset and analyzing summary statistics to understand the feature distributions and any potential issues such as skewed features or extreme outliers. The following operations were performed:

```
python
CopyEdit
# Load dataset and display basic statistics
print(data.head())
print(data.describe())
```

Key Observations:

- The target variable, **price**, had a wide range, suggesting significant variability in housing prices.
- Features such as **bedrooms** and **bathrooms** showed typical patterns, where more bedrooms correlated with larger living areas.

3.2 Missing Values

We checked for missing values across the dataset:

```
python
CopyEdit
print(data.isnull().sum())
```

Findings: There were minimal missing values, which were handled by filling them with the column mean. This approach was chosen to avoid discarding data that could be valuable for model training.

3.3 Correlation Analysis

To identify relationships between features and the target variable (**price**), we conducted a correlation analysis using a heatmap.

```
python
CopyEdit
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

Key Insights:

- **sqft_living** showed a strong positive correlation with **price**, meaning larger living spaces are likely associated with higher house prices.
- Features like **bedrooms** and **bathrooms** exhibited moderate correlation with price.
- **condition** and **yr_built** showed weaker correlations, suggesting that the overall condition and age of the house might not have a significant impact on the price.

---

4. Data Preprocessing

4.1 Handling Missing Values

Missing values were minimal but were still addressed by filling them with the mean of their respective columns:

```
python
CopyEdit
data.fillna(data.mean(), inplace=True
```
)

4.2 Encoding Categorical Features

Non-numeric features such as **street**, **city**, **statezip**, and **country** were encoded using one-hot encoding. This allows the model to handle categorical variables effectively:

```
python
CopyEdit
non_numeric_cols = data.select_dtypes(exclude=['number']).columns
data = pd.get_dummies(data, drop_first=True)
```

4.3 Feature Scaling

Feature scaling was applied to numerical features to standardize them, ensuring that no single feature disproportionately influences the model due to its scale:

```
python
CopyEdit
scaler = StandardScaler()
numerical_features = ['price', 'bedrooms', 'bathrooms', 'sqft_living']
data[numerical_features] = scaler.fit_transform(data[numerical_features])
```

5. Model Implementation and Training

5.1 Feature and Target Selection

We separated the dataset into features (**X**) and the target variable (**y**):

```
python
CopyEdit
X = data.drop(columns=['price'])
y = data['price']
```

5.2 Splitting the Data

To evaluate the model's performance, we split the data into a training set (80%) and a test set (20%):

```
python
CopyEdit
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

5.3 Training the Model

A **Linear Regression** model was trained using the training set:

```
python
CopyEdit
model = LinearRegression()
model.fit(X_train, y_train)
```

6. Model Evaluation

6.1 Predictions

After training the model, we made predictions on the test set:

```
python
CopyEdit
y_pred = model.predict(X_test)
```

6.2 Performance Metrics

The **Mean Squared Error (MSE)** was calculated to evaluate model performance. A lower MSE indicates better prediction accuracy:

```
python
CopyEdit
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
```

6.3 Visualization

To visually assess the model's performance, we plotted actual vs. predicted prices:

```
python
CopyEdit
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```

This scatter plot provides a visual indication of how well the model's predictions align with actual values.

7. Model Deployment

7.1 Saving the Model

Once the model was trained and evaluated, we saved it to disk for future use:

```
python
CopyEdit
import joblib
joblib.dump(model, 'model.pkl')
```

7.2 Deployment with FastAPI

A **FastAPI** application was created to serve the model. The API allows users to input property features and receive real-time predictions for house prices. Below is the FastAPI implementation:

```
python
CopyEdit
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import pandas as pd
import joblib

app = FastAPI()

class HouseFeatures(BaseModel):
    date: str
    price: float
    bedrooms: float
    bathrooms: float
    sqft_living: float
    sqft_lot: float
    floors: float
    waterfront: int
    view: int
    condition: int
    sqft_above: float
    sqft_basement: float
    yr_built: int
    yr_renovated: int
    street: str
    city: str
    statezip: str
    country: str

@app.post("/predict")
def predict_price(features: HouseFeatures):
    try:
        # Convert input to DataFrame
        input_data = pd.DataFrame([features.dict()])
        # Load the trained model
        model = joblib.load("model.pkl")
        # Make predictions
        prediction = model.predict(input_data.drop(columns=["date", "street",
"city", "statezip", "country"]))
        return {"predicted_price": prediction[0]}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

7.3 Running the API

The FastAPI application can be run using the following command:

```
bash
CopyEdit
uvicorn app:app --reload
```

7.4 Example Input and Output

Input:

```
json
CopyEdit
{
  "date": "2025-02-01",
  "price": 500000,
  "bedrooms": 3.0,
  "bathrooms": 2.0,
  "sqft_living": 1800.0,
  "sqft_lot": 5000.0,
  "floors": 2.0,
  "waterfront": 0,
  "view": 0,
  "condition": 3,
  "sqft_above": 1500.0,
  "sqft_basement": 300.0,
  "yr_built": 1995,
  "yr_renovated": 2010,
  "street": "123 Elm St",
  "city": "Seattle",
  "statezip": "WA 98101",
  "country": "USA"
}
```

Output:

```
json
CopyEdit
{
  "predicted_price": 480000.0
}
```

---

8. Results and Discussion

8.1 Model Strengths

- The model effectively captures linear relationships between the features and target variable.
- **Linear Regression** is a simple and interpretable model, making it easy to understand how predictions are derived.

8.2 Limitations

- The model might not capture complex, non-linear relationships between features and house prices, limiting its prediction accuracy in certain scenarios.
- The model is sensitive to outliers, which could distort the predictions.

---

9. Future Improvements

- **Feature Expansion:** Including more advanced features such as neighborhood ratings, proximity to schools, and other relevant factors could improve the model's performance.
- **Advanced Models:** Exploring more complex regression models, such as **Random Forest** or **Gradient Boosting**, could potentially yield better results.
- **Real-World Data:** Using a larger, real-world dataset would provide more diverse examples, improving the generalizability of the model.

---

10. Conclusion

This project successfully implemented a machine learning regression model to predict house prices. Through an extensive process of exploratory data analysis, preprocessing, model training, evaluation, and deployment, we demonstrated how machine learning can be applied to solve real-world problems in real estate. Future improvements will focus on incorporating more features, testing more advanced models, and working with real-world data to further enhance the predictive capabilities of the model.