

## 2. House Price Prediction (Linear Regression)

**Description:** Predict the price of a house based on features like area, number of bedrooms, location, etc.

Steps:

- Collect or use a dataset (like the Boston housing dataset or custom CSV).
- Clean and preprocess the data (handle missing values, encode categories).
- Use exploratory data analysis to understand correlations.
- Train a Linear Regression model.
- Evaluate performance using metrics like Mean Squared Error (MSE) or R<sup>2</sup> score

---

### Project Code

#### #main.py

```
# main.py
from src.utils import load_dataset
from src.data_preprocessing import preprocess_data
from src.model_training import train_model
from src.evaluation import evaluate_model

def main():
    # Load dataset
    df = load_dataset()

    # Preprocess data
    X_train, X_test, y_train, y_test, scaler = preprocess_data(df)

    # Train model
    model = train_model(X_train, y_train)

    # Evaluate
    results = evaluate_model(model, X_test, y_test)
    print("Model Evaluation Results:")
    print(results)

if __name__ == "__main__":
    main()
```

## # src/data\_preprocessing.py

```
# src/data_preprocessing.py
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

def preprocess_data(df):
    # Example: Drop missing values
    df = df.dropna()

    # Features and target
    X = df.drop("PRICE", axis=1)
    y = df["PRICE"]

    # Scaling features
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Split dataset
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42
    )
    return X_train, X_test, y_train, y_test, scaler
```

## # src/evaluation.py

```
# src/evaluation.py
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, predictions)
    return {"MSE": mse, "RMSE": rmse, "R2_Score": r2}
```

```
# src/evaluation.py
```

```
# src/evaluation.py
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, predictions)
    return {"MSE": mse, "RMSE": rmse, "R2_Score": r2}
```

```
# src/utils.py
```

```
# src/utils.py
import pandas as pd
from sklearn.datasets import fetch_california_housing

def load_dataset(use_csv=False, csv_path=None):
    if use_csv and csv_path:
        return pd.read_csv(csv_path)
    else:
        housing = fetch_california_housing(as_frame=True)
        df = housing.frame
        df.rename(columns={"MedHouseVal": "PRICE"}, inplace=True) # Rename target column
    return df
```

## Project Output

```
PS P:\CODE-XI\AI-Py-Learn\internship-spark-iit\tasks for ai\house_price_prediction> python -u "p:\CODE-XI\AI-Py-Learn\internship-spark-iit\tasks for ai\house_price_prediction\main.py"
Model Evaluation Results:
{'MSE': 0.5558915986952444, 'RMSE': np.float64(0.7455813830127764), 'R2_Score': 0.5757877060324508}
```