House Price Prediction - Project Documentation

HOUSE PRICE PREDICTION PROJECT - DOCUMENTATION

PROJECT OVERVIEW

This project predicts house prices using machine learning models.
It involves loading the dataset, analyzing features, splitting data into training and testing sets,
training the model using XGBoost Regressor, and evaluating performance using R scores.

DATA DESCRIPTION

The dataset consists of multiple housing-related features such as:
- Median Income (MedInc)
- House Age (HouseAge)
- Average Number of Rooms (AveRooms)
- Average Number of Bedrooms (AveBedrms)
- Population
- Average Occupancy (AveOccup)
- Latitude
- Longitude

The target variable is 'Price', representing the median house price.

WORKFLOW OVERVIEW

---

1. Importing necessary libraries
2. Loading and inspecting the dataset
3. Performing data preprocessing

4. Splitting the dataset into training and testing sets

5. Model training using XGBoost Regressor

6. Evaluating model performance using R , MSE, and MAE metrics

7. Visualizing correlation between features and price

MODEL USED

Model: XGBoost Regressor (XGBRegressor)

Description: An ensemble model based on Gradient Boosting that combines multiple weak learners (decision trees) to make strong predictions.

Key parameters:

- n_estimators: Number of trees
- learning_rate: Step size for each iteration
- max_depth: Maximum depth of trees
- random_state: Ensures reproducibility

CODE IMPLEMENTATION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics

house_price_dataset = sklearn.datasets.fetch_california_housing()
house_price_dataset

#Loading into pandas dataframe
```

```
house_price_dataframe = pd.DataFrame(house_price_dataset.data, columns=house_price_dataset.feature_names)
house_price_dataframe.head()

house_price_dataframe['price']=house_price_dataset.target

house_price_dataframe.head()

#checking the shape
house_price_dataframe.shape

#check missing values
house_price_dataframe.isnull().sum()

house_price_dataframe.describe()

correlation = house_price_dataframe.corr()

#construction heatmap to understand correlation
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap="Blues")

x = house_price_dataframe.drop(['price'], axis=1) #dataframe
y = house_price_dataframe['price'] #series



x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)

x.shape, x_train.shape, x_test.shape, y_train.shape, y_test.shape

#Model Training
```

```
#XGBoost Regressor

#Loading the model
model = XGBRegressor()

#training the model with x_train
model.fit(x_train, y_train)

#accuracy for prediction on training data
training_data_prediction = model.predict(x_train)
training_data_prediction

#Prediction on Training data
#R squared error
score_1 = metrics.r2_score(y_train, training_data_prediction)
print("R-squared value = ", score_1*100)

#Mean absolute error
score_2 = metrics.mean_absolute_error(y_train, training_data_prediction)
print("Mean Absolute Error = ",score_2) #near 0 means model is performing well

#So the accuracy is 94%

plt.scatter(y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Price vs Predicted Price")
plt.show()

#accuracy for prediction on test data
test_data_prediction = model.predict(x_test)
test_data_prediction
```

```
#Prediction on Test data
#R squared error
score_3 = metrics.r2_score(y_test, test_data_prediction)
print("R-squared value = ", score_3*100)

#Mean absolute error
score_4 = metrics.mean_absolute_error(y_test, test_data_prediction)
print("Mean Absolute Error = ",score_4) #near 0 means model is performing well
```