### **Over All Idea For Machine Learning**

### **Common Hypotheses in Housing Price Prediction Projects**

1. **Location Impact**: "The proximity to the city center (or coastline) significantly impacts housing prices."
2. **Housing Characteristics**: "Larger homes (more rooms, more square footage) command higher prices."
3. **Age of Property**: "Older properties tend to be cheaper than newer ones, unless they are renovated."

### **Addressing Each Hypothesis**

#### **Hypothesis 1: Location Impact**

* **Analysis Approach**:
  + Use the ocean\_proximity variable and any other geographical indicators available in your data.
  + Analyze average prices per location category and perform statistical tests (e.g., ANOVA) to determine if differences in means are statistically significant.
* **Visualizations**:
  + Plot average house prices by location category.
  + Show a map (if geographical coordinates are available) color-coded by price.

#### **Hypothesis 2: Housing Characteristics**

* **Analysis Approach**:
  + Correlate house size metrics (like total\_rooms, total\_bedrooms) with median\_house\_value.
  + Consider creating new features that could better capture the effects of size, such as rooms per household or bedrooms per room.
* **Visualizations**:
  + Scatter plots of room counts vs. house prices.
  + Box plots showing price distributions for different categories of house size.

#### **Hypothesis 3: Age of Property**

* **Analysis Approach**:
  + Correlate housing\_median\_age with median\_house\_value.
  + Create age categories (e.g., new, 10-20 years, over 20 years) and compare average prices among these categories.
* **Visualizations**:
  + Histogram or bar chart of house age vs. average prices.
  + Scatter plot with age as one axis and price as the other.

### **Answers to Other Questions**

**Q1: What factors predict housing prices the most accurately?**

* **Answer**: This can be addressed by looking at the feature importances output of the RandomForest model. Higher values indicate more predictive power.

**Q2: How do model predictions compare to actual housing prices?**

* **Answer**:
  + Use the scatter plot of actual vs. predicted values to visually assess this.
  + Calculate R-squared and MSE to quantify model performance.

**Q3: How does model complexity relate to prediction accuracy?**

* **Answer**:
  + Discuss the difference in performance between the simpler Linear Regression and the more complex RandomForest.
  + Use cross-validation results to talk about overfitting vs. underfitting.

Based on your findings:

* **Location-Based Investments**: If location significantly impacts prices, recommend focusing real estate investments in high-value areas.
* **Renovation Recommendations**: If older homes that are renovated bring higher prices, suggest focusing on renovations as a value-adding strategy.
* **Development Suggestions**: If larger homes command higher prices, recommend that developers focus on building or expanding properties in popular locations.

### **Data Cleaning and Preprocessing**

* **Missing Values**: housing\_data.isnull().sum() identifies missing values in each column, crucial for deciding how to handle them, as missing data can significantly affect model performance.
* **Correlation Matrix**: Exploring the relationships between numerical features with a correlation matrix helps identify potential predictors of the target variable and check for multicollinearity.

### **Feature Engineering and Preprocessing Pipeline**

* **Feature Identification**: Separating features into numeric and categorical lists to handle them appropriately during preprocessing.
* **Preprocessing Steps**:
  + **Numeric Transformer**: Imputes missing values and scales features.
  + **Categorical Transformer**: Imputes missing values and applies one-hot encoding to convert categorical variables into a machine-readable format.
* **Column Transformer**: Combines all preprocessing steps into a single transformer that can be applied to the dataset, ensuring that all features are appropriately preprocessed before modeling.

### **Model Building and Evaluation**

* **Data Splitting**: Dividing the data into training and testing sets is essential for training the model and then evaluating its performance on unseen data.
* **Model Training**: A linear regression model is trained on the preprocessed training data.
* **Prediction and Metrics**: The model is used to predict on the test set, and performance metrics (MSE and R²) are calculated to evaluate how well the model performs.

### **Visualization of Model Predictions**

* **Actual vs. Predicted Values Plot**: This scatter plot is crucial for visually assessing the accuracy of the predictions against the actual values.
* **Residual Plot**: Helps in understanding the distribution of residuals (errors), which can indicate issues with the model such as underfitting or bias.

### **Advanced Modeling with RandomForest**

* **Model Setup**: A more complex model, RandomForestRegressor, is used to potentially improve prediction accuracy due to its ability to model non-linear relationships and interactions between features.
* **Model Evaluation**: The RandomForest model is also evaluated using MSE and R², providing a comparison to see if a more complex model yields better performance.

### **Cross-Validation**

* **Robust Validation**: Applying cross-validation to assess the model’s performance more reliably across different subsets of the dataset, providing a better estimate of the model’s generalization ability.

**CODE WITH PROPER COMMENTS**

# Importing necessary libraries for data handling, visualization, and machine learning

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.impute import SimpleImputer

# Setting matplotlib to display plots inline in a Jupyter Notebook environment

%matplotlib inline

# Loading the housing data from a specified file path

file\_path = '/Users/borshapodder/Documents/Project \_Machine Learning/housing.csv'

housing\_data = pd.read\_csv(file\_path)

# Displaying the first few rows of the dataset to verify data loading

print(housing\_data.head())

# Printing basic info about the dataframe to understand the structure and data types

print(housing\_data.info())

# Printing summary statistics of the dataframe to understand the distribution of numerical data

print(housing\_data.describe())

# Creating histograms of all numerical features to visually inspect the distribution of data

housing\_data.hist(bins=50, figsize=(20, 15))

plt.show()

# Checking for missing values in the dataset to decide on further preprocessing needs

print("Missing values in each column:\n", housing\_data.isnull().sum())

# Selecting numeric columns only to prepare for correlation analysis

numeric\_cols = housing\_data.select\_dtypes(include=[np.number]) # Ensures only numeric columns are included

# Calculating the correlation matrix to understand relationships between features

correlation\_matrix = numeric\_cols.corr()

# Visualizing the correlation matrix using a heatmap to better understand feature interactions

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap='coolwarm')

plt.title('Correlation Matrix for Numeric Features')

plt.show()

# Identifying numeric and categorical features for preprocessing

numeric\_features = housing\_data.select\_dtypes(include=['int64', 'float64']).columns.tolist()

numeric\_features.remove('median\_house\_value') # Exclude the target variable from features

categorical\_features = ['ocean\_proximity']

# Creating pipelines for numerical and categorical feature preprocessing

numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='median')), # Fills missing values with the median

('scaler', StandardScaler()) # Standardizes features by removing the mean and scaling to unit variance

])

categorical\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='constant', fill\_value='missing')), # Fills missing values with 'missing'

('onehot', OneHotEncoder(handle\_unknown='ignore')) # Transforms categorical variable into dummy/indicator variables

])

# Combining preprocessing steps into a single ColumnTransformer

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)

])

# Defining the features and target variable for model training

X = housing\_data.drop('median\_house\_value', axis=1)

y = housing\_data['median\_house\_value']

# Splitting the dataset 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)

# Creating a pipeline that includes preprocessing and a linear regression model for prediction

model = Pipeline(steps=[

('preprocessor', preprocessor),

('regressor', LinearRegression())

])

# Training the model on the training data

model.fit(X\_train, y\_train)

# Making predictions on the test set

y\_pred = model.predict(X\_test)

# Calculating and printing evaluation metrics for model performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error (MSE): {mse}')

print(f'R^2 Score: {r2}')

# Visualizing actual vs predicted values to assess model accuracy

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--') # Line indicating perfect prediction

plt.xlabel('Actual Median House Value')

plt.ylabel('Predicted Median House Value')

plt.title('Actual vs. Predicted House Prices')

plt.show()

# Plotting residuals to check for patterns that might indicate problems with the model

residuals = y\_test - y\_pred

plt.figure(figsize=(10, 6))

sns.histplot(residuals, kde=True)

plt.xlabel('Residuals')

plt.title('Distribution of Residuals')

plt.show()

**Links: That I take help to understand and do the code for project**

* <https://stackoverflow.com/questions/60685866/how-do-i-find-out-the-rmse-of-a-random-forest-in-r>
* <https://medium.com/ampersand-academy/random-forest-regression-using-python-sklearn-from-scratch-9ad7cf2ec2bb>
* <https://www.kaggle.com/code/haseebwar07/random-forest-tuning-cross-validation>
* <https://stackoverflow.com/questions/71615078/how-to-do-cross-validation-on-random-forest>
* <https://www.kaggle.com/code/alexisbcook/pipelines>
* <https://www.w3resource.com/python-exercises/pandas_numpy/pandas_numpy-exercise-11.php#google_vignette>
* <https://www.geeksforgeeks.org/how-to-draw-2d-heatmap-using-matplotlib-in-python/>