LAB ON LINEAR REGRESSION

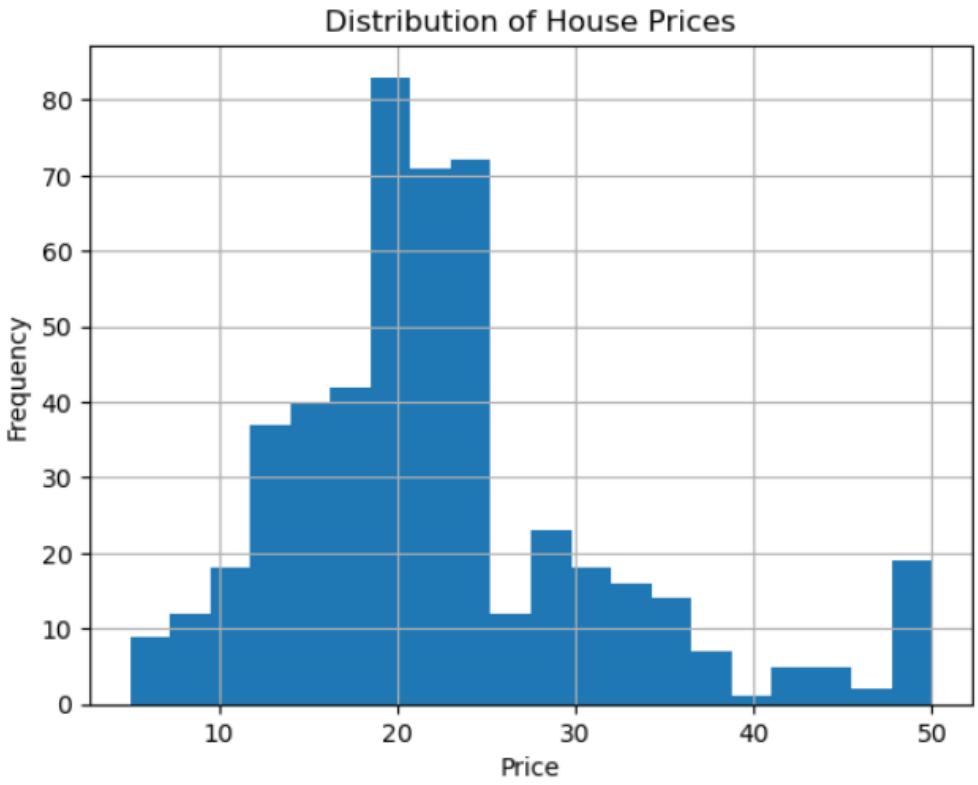
Lab – Linear Regression

- **Task A1: Data Loading and Exploration**
  - Load the Boston Housing dataset from sklearn
  - Create a DataFrame and examine its structure
    - How many samples and features are in the dataset?
    - What is the price range (min, max, mean)?
- **Task A2: Exploratory Data Analysis**
  - Create a histogram of house prices
  - Calculate *correlation matrix* between features and price
  - Create a scatter plot between price and the feature most correlated with price
    - Which feature has the strongest positive correlation with price?
    - Is the price distribution normal or skewed? What does this suggest?
- **Task A3: Model Building and Evaluation**
  - Split data into training (80%) and testing (20%) sets
  - Train a Linear Regression model
  - Make predictions on test set
  - Calculate and report R^2 score and RMSE -> **Research R^2 score**
  - Create scatter plot of actual vs predicted prices
    - What is your model's R^2 score? Is this considered good performance?
    - Based on the actual vs predicted plot, does your model perform better on certain price ranges?

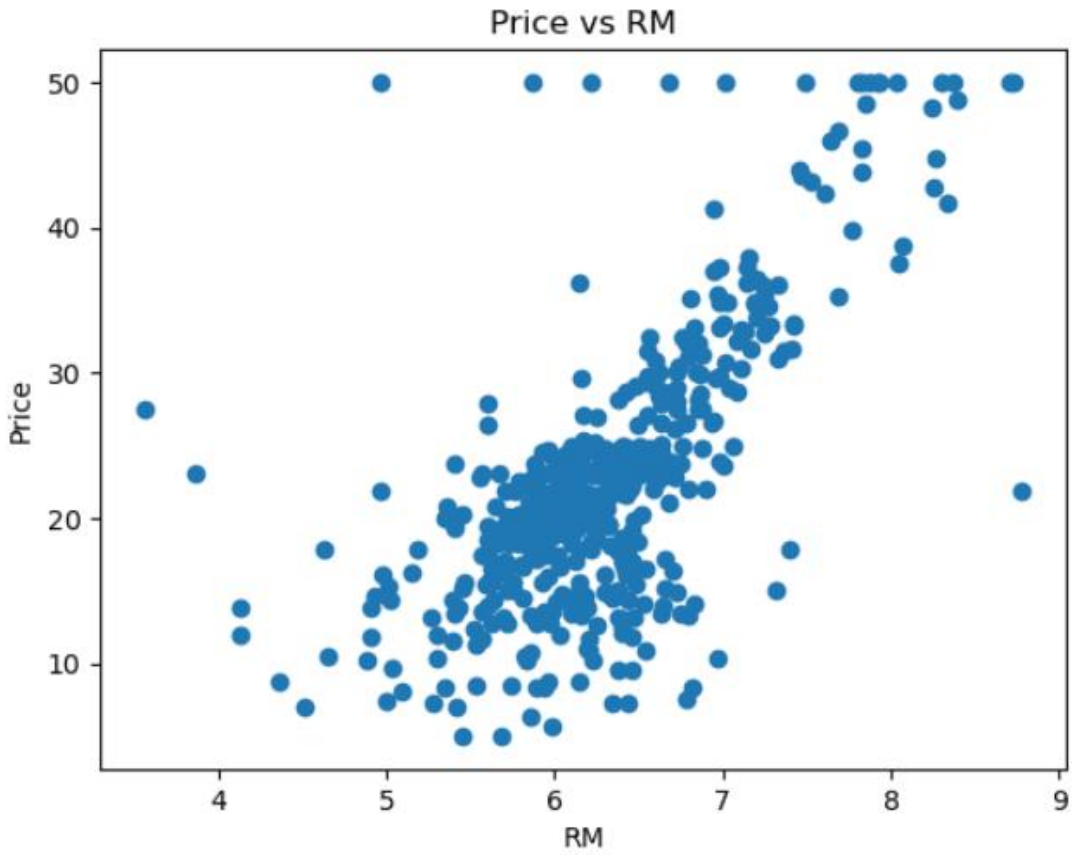
[1]:
```
import pandas as pd
import numpy as np

data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep=r"\s+", skiprows=22, header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]
feature_names = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD",
    "TAX", "PTRATIO", "B", "LSTAT"
]
df = pd.DataFrame(data, columns=feature_names)
df["PRICE"] = target

print(df.head())                  # See initial rows
print(df.shape)                   # Number of samples and features
print(df['PRICE'].min(), df['PRICE'].max(), df['PRICE'].mean())  # Price stats
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

   PTRATIO       B  LSTAT  PRICE
0     15.3  396.90   4.98   24.0
1     17.8  396.90   9.14   21.6
2     17.8  392.83   4.03   34.7
3     18.7  394.63   2.94   33.4
4     18.7  396.90   5.33   36.2
(506, 14)
5.0 50.0 22.532806324110677
```

[2]:
```
import matplotlib.pyplot as plt

# Histogram of prices
df['PRICE'].hist(bins=20)
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()

# Correlation matrix
corr = df.corr()
print(corr['PRICE'].sort_values(ascending=False))

# Strongest positive correlation feature (likely RM or similar)
top_feature = corr['PRICE'].drop('PRICE').idxmax()

print(f"Feature most positively correlated with Price: {top_feature}")

# Scatter plot price vs most correlated feature
plt.scatter(df[top_feature], df['PRICE'])
plt.xlabel(top_feature)
plt.ylabel('Price')
plt.title(f'Price vs {top_feature}')
plt.show()
```

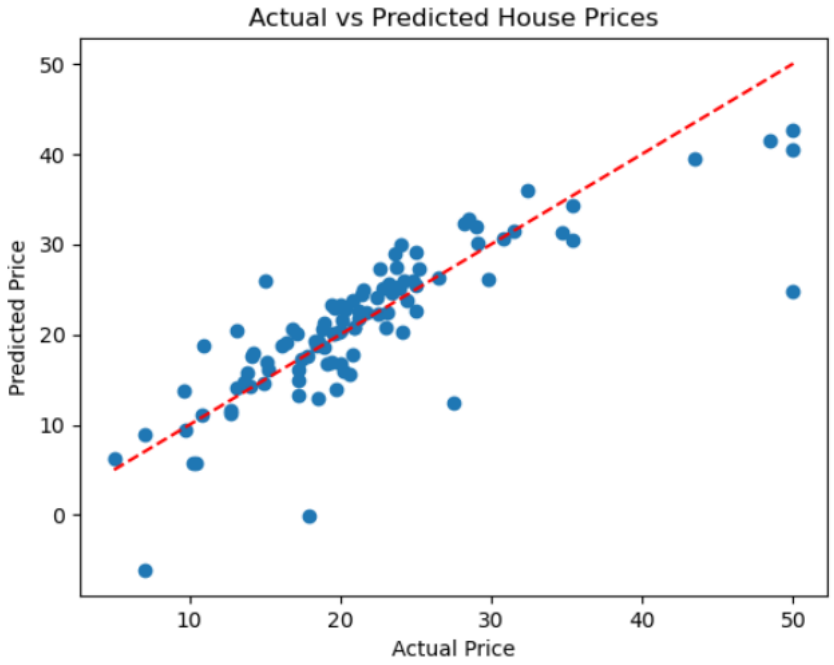


```
PRICE      1.000000
RM         0.695360
ZN         0.360445
B          0.333461
DIS        0.249929
CHAS       0.175260
AGE       -0.376955
RAD       -0.381626
CRIM      -0.388305
NOX       -0.427321
TAX       -0.468536
INDUS     -0.483725
PTRATIO   -0.507787
LSTAT     -0.737663
Name: PRICE, dtype: float64
Feature most positively correlated with Price: RM
```



[3]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

X = df.drop(columns="PRICE")
y = df["PRICE"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Metrics
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print(f"RMSE: {rmse:.2f}")
print(f"R^2: {r2:.2f}")

# Actual vs Predicted plot
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted House Prices")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')
plt.show()
```

```
RMSE: 4.93
R^2: 0.67
```



- R^2 score (Coefficient of Determination):
  This value indicates the proportion of variance in the housing prices that is predictable from the features.
  - Values range from 0 to 1, with values closer to 1 indicating a better fit.
  - For example, an R^2 score of 0.7 means 70% of the variance in prices is explained by the model features.
- RMSE (Root Mean Squared Error):
  This metric quantifies the average prediction error in the same units as the target (house prices).
  - Lower RMSE means predictions are closer to actual values.

- Scatter Plot of Actual vs Predicted Prices:
    - Points clustered near the diagonal line y=x$y=x$ indicate good predictive performance.
    - If the points spread widely or form patterns away from the line, it suggests underfitting or feature/model weaknesses.
- Distribution of Prices:
    - Check from histogram if prices are normally distributed or skewed.
    - Skewness suggests that simple linear models may struggle with predictions at extreme price ranges.
    - Consider data transformations or more advanced models if skewness is pronounced.
- Feature Impact Interpretation (Optional):
    - Features like 'RM' (average rooms) positively correlated with price means more rooms usually increase house value.
    - Features like 'LSTAT' (percentage lower status population) often negatively correlated, implying socioeconomic effects on house prices.