Regression

Uncovering the Patterns, Predicting the Future.

Avinash A S

1. Simple Linear Regression Model

```
# 1. Importing all the required Libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 2. Load the dataset
df = pd.read_csv("house_prices.csv")

# 3. Prepare the data for training
X = df[['Size']]  # Independent variable (features)
y = df['Price']   # Dependent variable (target)

# 4. Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 5. Create the linear regression model
model = LinearRegression()

# 6. Train the model using the training data
model.fit(X_train, y_train)

# 7. Make predictions on the test data
y_pred = model.predict(X_test)

# 8. Evaluate the model (using Mean Squared Error and R-squared)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# 9. Visualize the regression line
plt.scatter(df['Size'], df['Price'], color='blue', label='Data Points')
plt.plot(df['Size'], model.predict(df[['Size']]), color='red', label='Regression Line')
plt.title('Price vs Size (with Regression Line)')
plt.xlabel('Size (square feet)')
plt.ylabel('Price (dollars)')
plt.legend()
plt.show()

# 10. Predict the price for a new size (for example, 1500 square feet)
new_size = 1500
predicted_price = model.predict([[new_size]])
print(f"The predicted price for a house with {new_size} square feet is
{predicted_price[0]:,.2f}./- INR")
```

1. Importing Necessary Libraries

- **Pandas (pd)**: Used to load and handle the dataset in a DataFrame format.
- **Matplotlib (plt)**: Used for plotting graphs, here for visualizing the data points and regression line.
- **Train_test_split**: From sklearn.model_selection, used to split the dataset into training and testing sets.
- **LinearRegression**: From sklearn.linear_model, used to create and train a linear regression model.
- **Mean Squared Error (mean_squared_error)** and **R-squared (r2_score)**: From sklearn.metrics, used to evaluate the performance of the trained model.

2. Load the Dataset

- Load the house prices data from a CSV file into a DataFrame df.
- **Data:** The CSV file is assumed to have columns like Size (representing the size of the house in square feet) and Price (representing the price of the house).

3. Prepare Data for Training

- **X (Features):** We are using the Size column from the dataset as the independent variable, i.e., the feature used to predict the price.
- **y (Target):** We are using the Price column as the dependent variable, i.e., the value we want to predict.

4. Split the Data into Training and Testing Sets

- Split the data into two sets:
    - **Training set**: Used to train the model.
    - **Testing set**: Used to evaluate the model's performance after training.
- test_size=0.2: 20% of the data is reserved for testing.
- random_state=42: Ensures reproducibility of the data split (you will get the same split every time you run the code).

5. Create the Linear Regression Model

- Initialize a linear regression model. This model will be used to learn the relationship between house size (X) and house price (y).

6. Train the Model Using the Training Data

- Train the linear regression model using the training data (X_train and y_train).
- This step involves finding the best-fit line (the regression line) that minimizes the error between the predicted prices and the actual prices in the training set.

7. Make Predictions on the Test Data

- Use the trained model to predict the house prices (y_pred) for the test data (X_test).
- This will give us the predicted prices based on the model's understanding from the training data.

8. Evaluate the Model

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values. Lower values of MSE indicate better model performance.

- **R-squared (R^2):** A statistical measure that explains the proportion of the variance in the dependent variable that is predictable from the independent variable(s). R^2 values range from 0 to 1, with higher values indicating better model performance.
- These metrics help assess how well the model is performing on the test data.

9. Visualize the Regression Line

- Visualize the original data points (Size vs Price) and the regression line.
    - scatter() creates a scatter plot of the actual data points.
    - plot() overlays the regression line, which represents the model's predictions across all Size values.
    - **Color and Labels**: Helps in differentiating between the data points (blue) and the regression line (red).
    - show() displays the plot.

10. Predict the Price for a New Size

Make a prediction for a house with a new size (1500 square feet).

- **Input:** new_size = 1500 represents the new house size.
- **Prediction:** The predict() method is used to predict the price based on the model's learned relationship.
- **Print Result:** The predicted price for the house with 1500 square feet is displayed.

House Price Data Set(csv file):

```
Size,Price,Negotiable,Negotiable_%
400,130890,1,6
786,177200,1,3
1000,296785,1,8
1200,309999,0,0
1245,209800,0,0
1500,385689,1,2
1789,417567,0,0
1800,430123,0,5
2000,490780,1,5
2156,501200,0,0
2340,548890,1,4
2345,529078,1,6.5
2871,684200,0,0
3120,694000,1,5.6
3150,688976,1,4.8
890,212500,1,7
1504,375400,1,2.5
1805,450100,0,3
2050,499999,1,4.2
2243,535600,1,6
2450,556900,0,0
2600,589300,1,5.5
2785,652700,1,3.2
3000,700150,0,0
3300,720400,1,6.8
```

2. Mutli Regression Model

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Create the dataset
data = pd.read_csv("house_prices.csv")

# Convert the data into a DataFrame
df = pd.DataFrame(data)

# Define independent variables (features) and dependent variable (target)
X = df[['Size', 'Negotiatable', 'Negotiatable_%']]  # Independent variables
y = df['Price']  # Dependent variable

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the linear regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)

# Coefficients and intercept of the trained model
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Display the predicted prices
print("Predicted Prices:", y_pred)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate R-squared (R²)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Custom input for prediction (No Price value, only features: Size, Negotiatable,
Negotiatable_%)
custom_input = [[1867, 1, 2]]  # Example: Size = 1867 sq ft, Negotiatable = 1,
Negotiatable_% = 2%

# Make prediction for the custom input
custom_prediction = model.predict(custom_input)

print(f"Predicted Price for custom input : {custom_prediction[0]}")
```

1. Importing Necessary Libraries

- **Pandas (pd)**: Used for data manipulation and analysis. Here, it's used to read the dataset from a CSV file.
- **LinearRegression**: A class from sklearn.linear_model that provides tools for performing linear regression.
- **train_test_split**: From sklearn.model_selection, it splits the dataset into training and testing sets.
- **mean_squared_error, r2_score**: From sklearn.metrics, they help in evaluating the model's performance using metrics like MSE and R-squared.

2. Load the Dataset

- Loads the dataset from a CSV file named house_prices.csv into the data variable.
- **Assumption:** The CSV file contains data for house prices, including columns like Size, Negotiatable, Negotiatable_%, and Price.

3. Convert the Data into a DataFrame

- Converts the raw data into a DataFrame (a 2D labeled structure) for easy manipulation, especially when working with the data using libraries like pandas.

4. Define Independent and Dependent Variables

- **Independent Variables (X)**: We select the Size, Negotiatable, and Negotiatable_% columns as the features or predictors for the model.
- **Dependent Variable (y)**: The target variable is the Price column, which the model will attempt to predict based on the features.

5. Split the Data into Training and Testing Sets

- Split the dataset into training (80%) and testing (20%) sets.
    - **X_train, y_train**: Training data for the features and the target.
    - **X_test, y_test**: Testing data for features and target.
- **test_size=0.2**: Specifies that 20% of the data will be reserved for testing.
- **random_state=42**: Ensures reproducibility of the data split by setting a seed.

6. Initialize and Train the Linear Regression Model

- Create an instance of the LinearRegression model and then fit the model to the training data (X_train, y_train). The model learns the relationship between the independent variables (Size, Negotiatable, Negotiatable_%) and the dependent variable (Price).

7. Coefficients and Intercept of the Model

- **Coefficients:** The weights assigned to each independent variable in the linear regression model. These values represent how much change in the Price is expected for a unit change in each feature.
- **Intercept:** The predicted value of Price when all independent variables are zero (the value at the origin of the regression line).

8. Make Predictions on the Test Set

- Use the trained model to make predictions (y_pred) on the test set (X_test).

9. Display Predicted Prices

- Print the predicted house prices for the test data (X_test).

10. Evaluate the Model

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted prices. Lower values of MSE indicate a better fit of the model.
- **R-squared (R^2):** A measure of how well the independent variables explain the variation in the dependent variable. An R^2 value closer to 1 indicates a better fit.

11. Custom Input for Prediction

- Provide a custom input for prediction.
    - custom_input: A new set of data where the house has a size of 1867 square feet, it is negotiable, and the negotiability percentage is 2%.
    - **Prediction:** The predict() method generates the predicted price for this new input.
    - **Print Result:** Outputs the predicted price for the given custom input.

3. Polynomial Regression

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Create the dataset
data = pd.read_csv("house_prices.csv")

# Convert the data into a DataFrame
df = pd.DataFrame(data)

# Define independent variable (feature) and dependent variable (target)
X = df[['Size']]  # Independent variable (Size)
y = df['Price']   # Dependent variable (Price)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Transform the features to polynomial features (degree 3 for cubic)
poly = PolynomialFeatures(degree=3)
X_poly_train = poly.fit_transform(X_train)
X_poly_test = poly.transform(X_test)

# Initialize the linear regression model
model = LinearRegression()

# Train the model using the transformed training data
model.fit(X_poly_train, y_train)

# Coefficients and intercept of the trained model
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Make predictions on the test set
y_pred = model.predict(X_poly_test)

# Display the predicted prices
print("Predicted Prices (in INR):", y_pred)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate R-squared (R²)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Visualize the results
plt.scatter(X, y, color='blue')  # Original data
plt.plot(X, model.predict(poly.transform(X)), color='red')  # Polynomial regression line
plt.title('Polynomial Regression (Degree 3)')
plt.xlabel('Size (sq ft)')
```

```
plt.ylabel('Price (INR)')
plt.show()

# Custom input for prediction
custom_input = [[1867]]  # Example: Size = 1867 sq ft

# Make prediction for the custom input
custom_prediction = model.predict(poly.transform(custom_input))

print(f"Predicted Price for custom input (Size = 1867 sq ft):
₹{custom_prediction[0]:,.2f}")
```

4. Polynomial Plant Growth Data Regression:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# 1. Load the dataset
data = pd.read_csv("regression\plant_growth_data.csv")

# 2. Check the first few rows of the dataset
print(data.head())

# 3. Define independent and dependent variables
X = data[['Time']]  # Independent variable (Time)
y = data['Height']  # Dependent variable (Height)

# 4. Polynomial transformation (degree 4 for cubic relationship)
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X)

# 5. Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,
random_state=42)

# 6. Fit the Polynomial Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# 7. Predict on test data
y_pred = model.predict(X_test)

# 8. Model coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# 9. Calculate Mean Squared Error and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# 10. Plot the original data and the polynomial fit
plt.scatter(data['Time'], data['Height'], color='blue', label='Original Data')  #
Original data points
plt.plot(data['Time'], model.predict(poly.transform(data[['Time']])), color='red',
label='Polynomial Fit')  # Polynomial regression line
plt.title('Plant Growth Over Time')
plt.xlabel('Time (days)')
plt.ylabel('Height (cm)')
plt.legend()
plt.show()
```

```
# 11. Custom input for prediction (Predicting height at day 25)
custom_input = [[25]]  # Predicting the height at day 25
custom_input_poly = poly.transform(custom_input)
custom_prediction = model.predict(custom_input_poly)
print(f"Predicted Height at day 25: {custom_prediction[0]:.2f} cm")
```

https://github.com/avinashsatalagaon/Data-Analytics_Requirements/tree/main