Hands-on Lab: Final Project: Generative AI for Data Science

Estimated Effort: 60 mins

Project Scenario

You have been employed as a Data Scientist by a consultancy firm. The firm has a client who is a used car dealer. They have a special feature on Ford cars and they want your firm to design a model that can predict the optimum quotation price for the cars in their lot. They provide you with sales data for the past few years. The dataset contains different features of the cars and the price they were sold at.

The tasks assigned to you are as follows.

1. There might be a few duplicate entries and a few missing values in the dataset. Data cleaning will be a part of the assignment.
2. You have to perform exploratory data analysis to draw keen insights on the data and determine the effect of different features on the price. Some specific requests by the client include: a. Identify number of sales for each fuel type
   b. Identify which transmission type has more price outliers
3. Compare the models with linear, polynomial and ridge regressions on single and multiple variables to find the best performing model
4. Perform a Grid Search on the Ridge regression model to identify the optimum hyperparameter for the model for best performance.

You decide to use Generative AI to create python codes that can help you analyse the data, determine the best features and create the prediction model as per requirement.

> Disclaimer: This is a fictitious scenario created for the purpose of this project. The dataset being used is publicly available.

About the Dataset

This dataset contains used car sale prices for Ford cars. This is a public dataset available on the Kaggle website as Ford Car Pricing Dataset under the CC0: Public Domain license. The dataset has been slightly modified for the purpose of this project.

Attributes of this dataset have been explained below.

Variable	Description
model	Car model name
year	Year of car make
transmission	Type of transmission (Automatic, Manual or Semi-Auto)
mileage	Number of miles traveled
fuelType	The type of fuel the car uses (Petrol, Diesel, Hybrid, Electric, Other)
tax	Annual Tax payable in USD
mpg	Miles per Gallon that the car runs at
engineSize	Engine Size of the car
price	Price of car in USD

Code execution environment

To test the prompt-generated code, keep the Jupyter Notebook (in the link below) open in a separate tab in your web browser. The notebook has some setup instructions that you should complete now.

Jupyter-Lite Test Environment

> Please note the lab environment above will only work on Windows (Google Chrome or Firefox browser). If you don't have a Windows system with either of these browsers, use the lab environment provided in the next lesson of the module.

The data set for this lab is available in the following URL.

URL = https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0271EN-SkillsNetwork/labs/v1/m3/data/used_car_price_analysis.csv

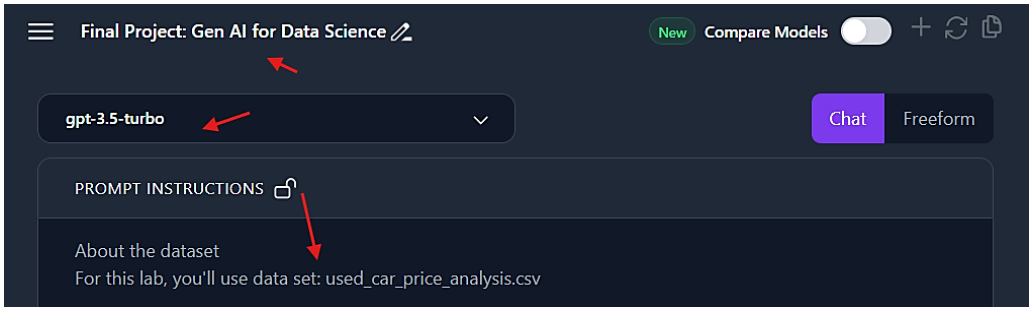
```
1  URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0271EN-SkillsNetwork/labs/v1/m3/data/used_car_price_analysis.
```

Complete the setup in the Jupyter Notebook and then proceed further.

> Important Note: All prompts that are made available have been hidden and the users are encouraged to first try to write their own prompts to create the solutions. Also, the prompts given as solutions have also been maintained as ones which will create generic code structures which you can modify according to the question at hand.

Importing the Dataset

You can begin by using the Generative AI model to create a python script that can load the dataset to a pandas dataframe. The dataset file already has the headers in the first row.



Write the prompt to generate the said code and test it in the JupyterLite environment. For verification of appropriate loading, include a step for printing the first 5 values of the loaded dataframe. ▼

Click here for the prompt

```
Write a Python code that can perform the following tasks.
1. Read the CSV file, located on a given file path, into a pandas data frame, assuming that the first row of the file can be used as the headers for the data.
2. Print the first 5 rows of the dataframe to verify correct loading.
```

```
1  Write a Python code that can perform the following tasks.
2  1. Read the CSV file, located on a given file path, into a pandas data frame, assuming that the first row of the file can be used as the headers for the data.
3  2. Print the first 5 rows of the dataframe to verify correct loading.
```

Write a Python code that can perform the following tasks.
1. Read the CSV file, located on a given file path, into a pandas data frame, assuming that the first row of the file can be used as the headers for the data.
2. Print the first 5 rows of the dataframe to verify correct loading.

This Python code reads a CSV file into a pandas data frame and then prints the first 5 rows of the dataframe to verify correct loading. Make sure to replace 'path_to_your_file/used_car_price_analysis.csv' with the actual file path where the CSV file is located.

```python
import pandas as pd

# Specify the file path
file_path = 'path/to/used_car_price_analysis.csv'

# Read the CSV file into a pandas data frame
df = pd.read_csv(file_path)

# Print the first 5 rows of the dataframe
print(df.head())
```

Data Preparation

Data Cleaning

At this stage, it is required to clean up the data. As has been informed to you, the data may have missing values and duplicate entries. Write a prompt that performs the following tasks

1. Identifies the columns with missing values and fills the blank cells with average value of the columns.
2. Identifies and drops the duplicate entries from the data.

▼ Click here for prompt

```
Write a python code that performs the following tasks:
1. Identify the columns of a data frame with missing values.
2. Replace the missing values thus identified with mean value of the column.
3. Deletes the duplicate entries from the data frame.
```

```
Write a python code that performs the following tasks:
1. Identify the columns of a data frame with missing values.
2. Replace the missing values thus identified with mean value of the column.
3. Deletes the duplicate entries from the data frame.
```

Write a python code that performs the following tasks:
1. Identify the columns of a data frame with missing values.
2. Replace the missing values thus identified with mean value of the column.
3. Deletes the duplicate entries from the data frame.

This Python code defines a function that identifies columns with missing values in a dataframe, replaces the missing values with the mean of the column, and then deletes duplicate entries from the dataframe.

```python
import pandas as pd

# Assume df is the pandas data frame

# Identify columns with missing values
columns_with_missing_values = df.columns[df.isnull().any()]

# Replace missing values with mean of the column using explicit assignment
for column in columns_with_missing_values:
    df.fillna({column: df[column].mean()}, inplace=True)

# Delete duplicate entries
df.drop_duplicates(inplace=True)
```

Data Augmentation (optional)

Once cleaned, you may choose to augment this dataset with additional samples, created synthetically using Mostly.ai.

```python
df.to_csv('cleaned_data.csv', index=False)
```

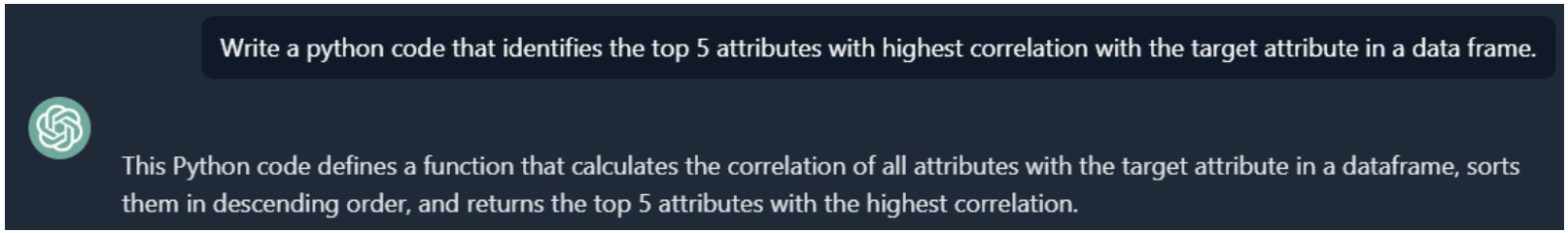
Data Insights and Visualization

Write **prompts** that generate codes to perform the following actions.

1. Identify the **5 attributes** that have the **highest correlation** with the **price parameter**.

▼ Click here for the prompt

```
Write a python code that identifies the top 5 attributes with highest correlation with the target attribute in a data frame.
```




```python
import pandas as pd

# Assume df is the pandas data frame

# Select only numeric columns
numeric_df = df.select_dtypes(include='number')

# Calculate correlations with the target attribute
target_attribute = 'target'
correlations = numeric_df.corr()[target_attribute].abs().sort_values(ascending=False)
top_5_attributes = correlations[1:6]
print(top_5_attributes)
```

2. **Count** the number of cars under each **unique** value of **fuelType attribute**.

▼ Click here for the prompt

```
Write a python code that counts the number of entries in a dataframe with each unique value of a specific attribute.
```

```python
import pandas as pd

# Assume df is the pandas data frame

# Specify the attribute for which unique value counts are needed
specific_attribute = 'attribute_name'

# Count the number of entries for each unique value of the specific attribute
value_counts = df[specific_attribute].value_counts()
print(value_counts)
```

3. Create a **Box plot** to determine whether cars with **automatic**, **manual** or **semi-auto type** of **transmission** have more **price outliers**. Use the **Seaborn** library for creating the plot.

▼ Click here for the prompt

```
1  Write a python code that creates a Seaborn box plot between a source and target attributes of a dataframe.
```

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Assume df is the pandas data frame
sns.boxplot(x='source_attribute', y='target_attribute', data=df)
plt.show()
```

4. Generate the **regression plot** between **mpg parameter** and the **price** to determine the **correlation type** between the **two**.

▼ Click here for the prompt

```
1  Write a python code that creates a Seaborn box plot between a source and target attributes of a dataframe.
```

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Assume df is the pandas data frame

# Create a regression plot between a source attribute and a target attribute
sns.regplot(x='source_attribute', y='target_attribute', data=df)
plt.show()
```

Model Development and Evaluation

Write prompts that generate codes to perform the following actions.

1. **Fit** a **linear regression model** to predict the **price** using the **feature mpg**. Then calculate the **R^2** and **MSE values** for the model.

▼ Click here for the prompt

```
1  Write a python code to fit a linear regression model on a target variable on basis of a source variable. Calculate the R^2 values and MSE values for this model.
```

Write a python code to fit a linear regression model on a target variable on basis of a source variable. Calculate the R^2 values and MSE values for this model.

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

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable']]
y = df['target_variable']

# Fit a linear regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions
y_pred = model.predict(X)

# Calculate R^2 and MSE
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

2. Fit a **linear regression model** to predict the `price` using the following **set** of **features**.
year, **mileage**, **tax**, **mpg** and **engineSize**.
Calculate the **R^2** and **MSE** values for this **model**.

▼ Click here for the prompt

```
Write a python code to fit a linear regression model on a target variable on basis of a set of source variables. Calculate the R^2 values and MSE values for this model.
```

Write a python code to fit a linear regression model on a target variable on basis of a set of source variables. Calculate the R^2 values and MSE values for this model.

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

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable_1', 'source_variable_2', 'source_variable_3']]  # Add all relevant source variables
y = df['target_variable']

# Fit a linear regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions
y_pred = model.predict(X)

# Calculate R^2 and MSE
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

3. For the **same set** of **features** as in the question above, create a **pipeline model** object that uses **standard scalar**, **second degree polynomial features** and a **linear regression model**.

Calculate the **R^2** value and the **MSE value** for this **model**.

▼ Click here for the prompt

Write a **python code** that can perform the following tasks.

```
Write a python code that can perform the following tasks.

1. Create a pipeline with standard scalar, second degree polynomial features and linear regression model.
2. Fit this pipeline for a target variable using a set of source attributes from a dataframe.
3. Evaluate the R^2 and MSE values for the trained model.
```

Write a python code that can perform the following tasks.
1. Create a pipeline with standard scalar, second degree polynomial features and linear regression model.
2. Fit this pipeline for a target variable using a set of source attributes from a dataframe.
3. Evaluate the R^2 and MSE values for the trained model.

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import pandas as pd

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable_1', 'source_variable_2', 'source_variable_3']]  # Add all relevant source variables
y = df['target_variable']

# Create a pipeline with standard scalar, polynomial features, and linear regression
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly_features', PolynomialFeatures(degree=2)),
    ('linear_reg', LinearRegression())
])

# Fit the pipeline
pipeline.fit(X, y)

# Make predictions
y_pred = pipeline.predict(X)

# Calculate R^2 and MSE
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

4. For the same set of features, split the data into **training** and **testing** data parts. Assume testing part to be **20%**. Create and **fit** a **Ridge regression** object using the training data, set the **regularization parameter** to **0.1**, and calculate the **R^2** using the **test** data.

▼ Click here for the prompt

```
Write a python code that can perform the following tasks.

1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Create and fit a Ridge regression model using the training data, setting the regularization parameter to 0.1.
3. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.
```

Write a python code that can perform the following tasks.
1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Create and fit a Ridge regression model using the training data, setting the regularization parameter to 0.1.
3. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.

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

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable_1', 'source_variable_2', 'source_variable_3']]  # Add all relevant source variables
y = df['target_variable']

# Split the data 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)

# Create and fit a Ridge regression model
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = ridge_model.predict(X_test)

# Calculate R^2 and MSE for the Ridge Regression model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

5. Perform a **second order polynomial transform** on both the **training** data and **testing** data created for the question above. Create and **fit** a **Ridge regression object** using the modified training data, set the **regularisation** parameter to **0.1**, and calculate the **R^2** and **MSE** utilising the **modified test** data.

▼ Click here for the prompt

```
Write a python code that can perform the following tasks.

1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Apply second degree polynomial scaling to the training and testing data.
3. Create and fit a Ridge regression model using the training data, setting the regularization parameter to 0.1.
4. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.
```

Write a python code that can perform the following tasks.
1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Apply second degree polynomial scaling to the training and testing data.
3. Create and fit a Ridge regression model using the training data, setting the regularization parameter to 0.1.
4. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score, mean_squared_error
import pandas as pd

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable_1', 'source_variable_2', 'source_variable_3']]  # Add all relevant source variables
y = df['target_variable']

# Split the data 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)

# Apply second degree polynomial scaling to the training and testing data
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Create and fit a Ridge regression model
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(X_train_poly, y_train)

# Make predictions on the testing data
y_pred = ridge_model.predict(X_test_poly)

# Calculate R^2 and MSE for the Ridge Regression model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

6. In the question above, perform a **Grid Search** on **ridge regression** for a set of **values** of **alpha** {**0.01, 0.1, 1, 10, 100**} with **4-fold cross validation** to find the optimum value of **alpha** to be used for the **prediction model**.

▼ Click here for the prompt

```
Write a python code that can perform the following tasks.

1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Apply second degree polynomial scaling to the training and testing data.
3. Create and fit a Grid search on Ridge regression with cross validation using the training data, for a set of values of the parameter alpha.
4. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.
```

Write a python code that can perform the following tasks.
1. Assuming that a subset of the attributes of a data frame are source attributes and one of the attributes is a target attribute, split the data into training and testing data assuming the testing data to be 20%.
2. Apply second degree polynomial scaling to the training and testing data.
3. Create and fit a Grid search on Ridge regression with cross validation using the training data, for a set of values of the parameter alpha.
4. Calculate the MSE and R^2 values for the Ridge Regression model using the testing data.

```python
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score, mean_squared_error
import pandas as pd

# Assume df is the pandas data frame

# Define source and target variables
X = df[['source_variable_1', 'source_variable_2', 'source_variable_3']]  # Add all relevant source variables
y = df['target_variable']

# Split the data 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)

# Apply second degree polynomial scaling to the training and testing data
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Create Ridge regression model
ridge = Ridge()
parameters = {'alpha': [0.1, 1, 10]}  # Define set of alpha values to search

# Perform Grid Search with Cross Validation
ridge_cv = GridSearchCV(ridge, parameters, cv=5)
ridge_cv.fit(X_train_poly, y_train)

# Make predictions on the testing data
y_pred = ridge_cv.predict(X_test_poly)

# Calculate R^2 and MSE for the Ridge Regression model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R^2 Score:", r2)
print("Mean Squared Error:", mse)
```

Conclusion

Congratulations! You have completed this guided project on using Generative AI for different data science tasks.

By the end of this project, you are now capable of using Generative AI for the tasks of:

- **Data preparation**: cleaning, transforming and augmentation
- **Data analysis**: drawing insight, creating visualizations
- **Model development**: creating simple as well as complex prediction models
- **Model refinement**: found the optimum model using Grid Search

Author(s)

Abhishek Gagneja