Skills Network

Hands-on Lab: Generative AI for Data Analysis and Mining

Estimated Effort: 30 mins

Introduction

One of the final tasks performed by a data engineer is to analyze the final data, draw insights from it, and employ data mining strategies to extract hidden patterns in the data distribution. In this lab, you will learn how to use generative AI for creating Python codes that can perform the required data analysis and data mining strategies.

Scenario

As a senior data engineer for a healthcare company, you are tasked to perform data analysis and data mining on patients' health records indicating whether or not the patient has been identified with a liver disease or not. Other teams have recorded and cleaned the data that is ready for analysis.

Objectives

In this lab, you will learn how to use generative AI to:

1. Perform exploratory data analysis on a given data set.
2. Perform data mining on the given data set and draw insights from the data.

Data set

For the purpose of this lab, we are making use of the Indian Liver Patient Dataset, publically available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. You may refer to the data set web page for more details on the attributes.

The data set is available for use in this lab at the following URL:

```
URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/ILPD.csv"
```

Copied!

Testing Interface

You will find a separate testing interface at the end of the lesson on the course page. Please keep that testing interface open as a separate lab on the side and follow the initial steps to be ready with the setup.

Exploratory Data Analysis

With exploratory data analysis, you can identify the type of data distribution and how different variables affect each other. In this task, you will use generative AI to create a code that does the following:

1. Create a histogram plot of the data based on the 'Age', 'Gender' and 'Selector' attributes.
2. Create a correlation heat map for the data set.
3. Select the top 5 features with the highest correlation value with the 'Selector' attribute.

   Note: Save the generated graphs to the local machine to be viewed. You cannot open images in this platform.

You can use the GenAI classroom with the following prompt:

```
For a given data set available in a CSV format, write a python code that does the following:
1. Load the CSV data to a pandas data frame assuming the first row of the CSV contains the headers of the data.
2. Save an image to disk, containing histograms of the data distribution with respect to a continuous attribute 'Age' and discrete attributes 'Gender
3. Save an image to disk containing the correlation heatmap of the data set.
4. Identify the top 5 features with highest absolute correlation with the attribute `Selector`.
```

Copied!

You can expect the generated code to be similar to the one shared below.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the CSV data into a pandas data frame
data = pd.read_csv('data.csv')

# Save histograms of data distribution for 'Age', 'Gender', and 'Selector'
plt.figure()
data['Age'].hist()
plt.savefig('age_histogram.png')

plt.figure()
data['Gender'].hist()
plt.savefig('gender_histogram.png')

plt.figure()
data['Selector'].hist()
plt.savefig('selector_histogram.png')

# Save correlation heatmap of the data set
plt.figure()
corr = data.corr()
sns.heatmap(corr, annot=True)
plt.savefig('correlation_heatmap.png')

# Identify top 5 features with highest absolute correlation with 'Selector'
correlation_with_selector = corr['Selector'].abs().sort_values(ascending=False)
top_5_features = correlation_with_selector[1:6]
# Exclude 'Selector' itself
print(top_5_features)
```

Copied!

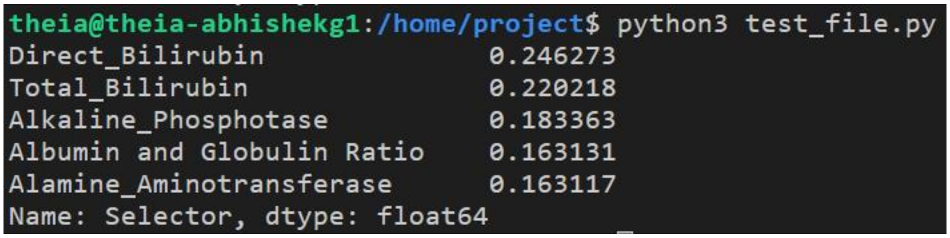
You can edit the URL, adjust a few lines of the code to make the output a little more usable, and the final code should look as shown below.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the CSV data into a pandas data frame
data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/ILPD.c

# Save histograms of data distribution for 'Age', 'Gender', and 'Selector'
plt.figure()
data['Age'].hist()
plt.savefig('age_histogram.png')

plt.figure()
data['Gender'].hist()
plt.savefig('gender_histogram.png')

plt.figure()
data['Selector'].hist()
plt.savefig('selector_histogram.png')

# Save correlation heatmap of the data set
plt.figure(figsize=(12,8))
corr = data.corr()
sns.heatmap(abs(corr), annot=True)
plt.savefig('correlation_heatmap.png', bbox_inches='tight')

# Identify top 5 features with highest absolute correlation with 'Selector'
correlation_with_selector = corr['Selector'].abs().sort_values(ascending=False)
top_5_features = correlation_with_selector[1:6]
# Exclude 'Selector' itself
print(top_5_features)
```

Copied!

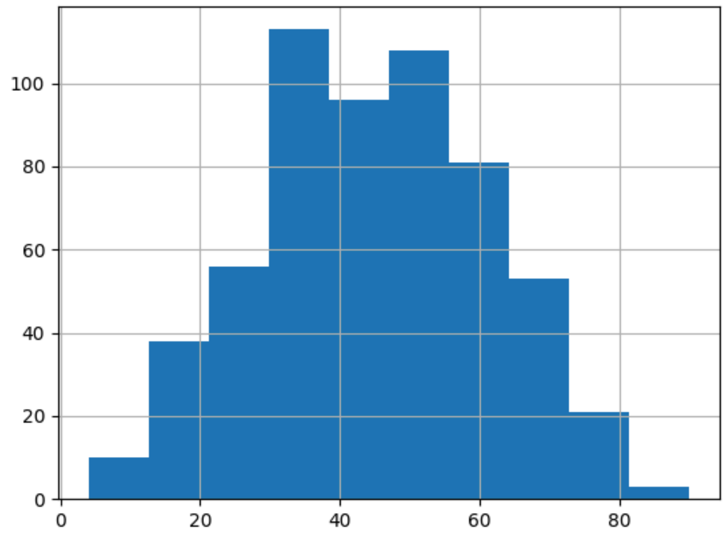
The required outputs would be as shown below.

1. Terminal Output

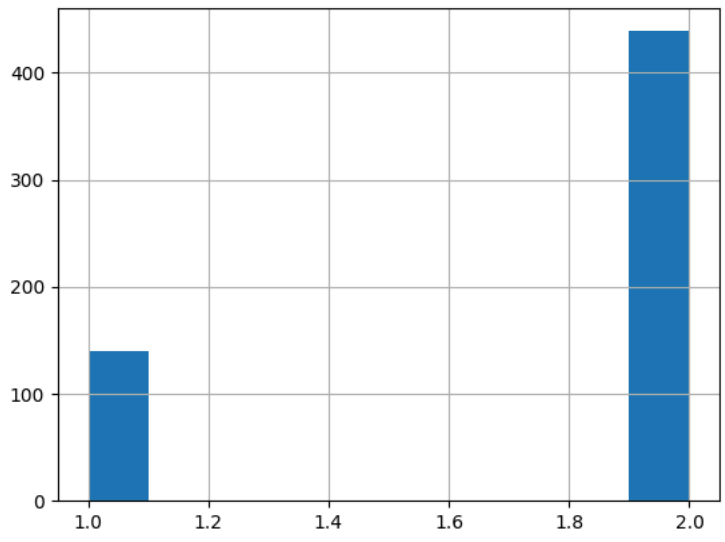


```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
Direct_Bilirubin              0.246273
Total_Bilirubin               0.220218
Alkaline_Phosphotase          0.183363
Albumin and Globulin Ratio    0.163131
Alamine_Aminotransferase      0.163117
Name: Selector, dtype: float64
```

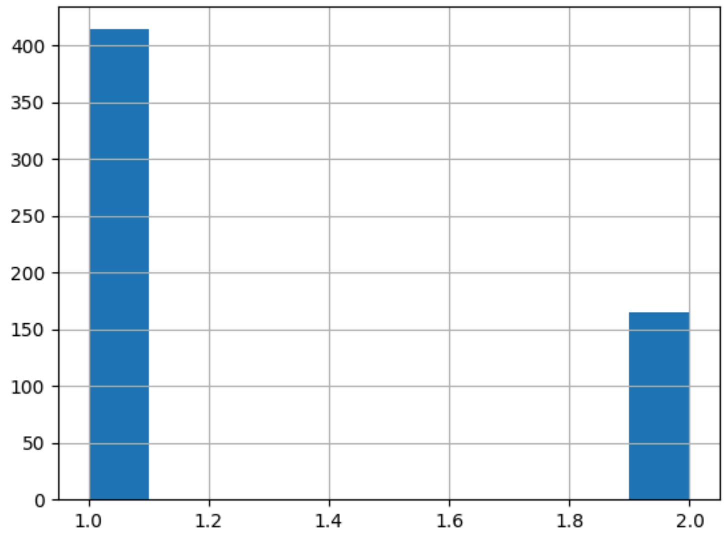
2. Age histogram



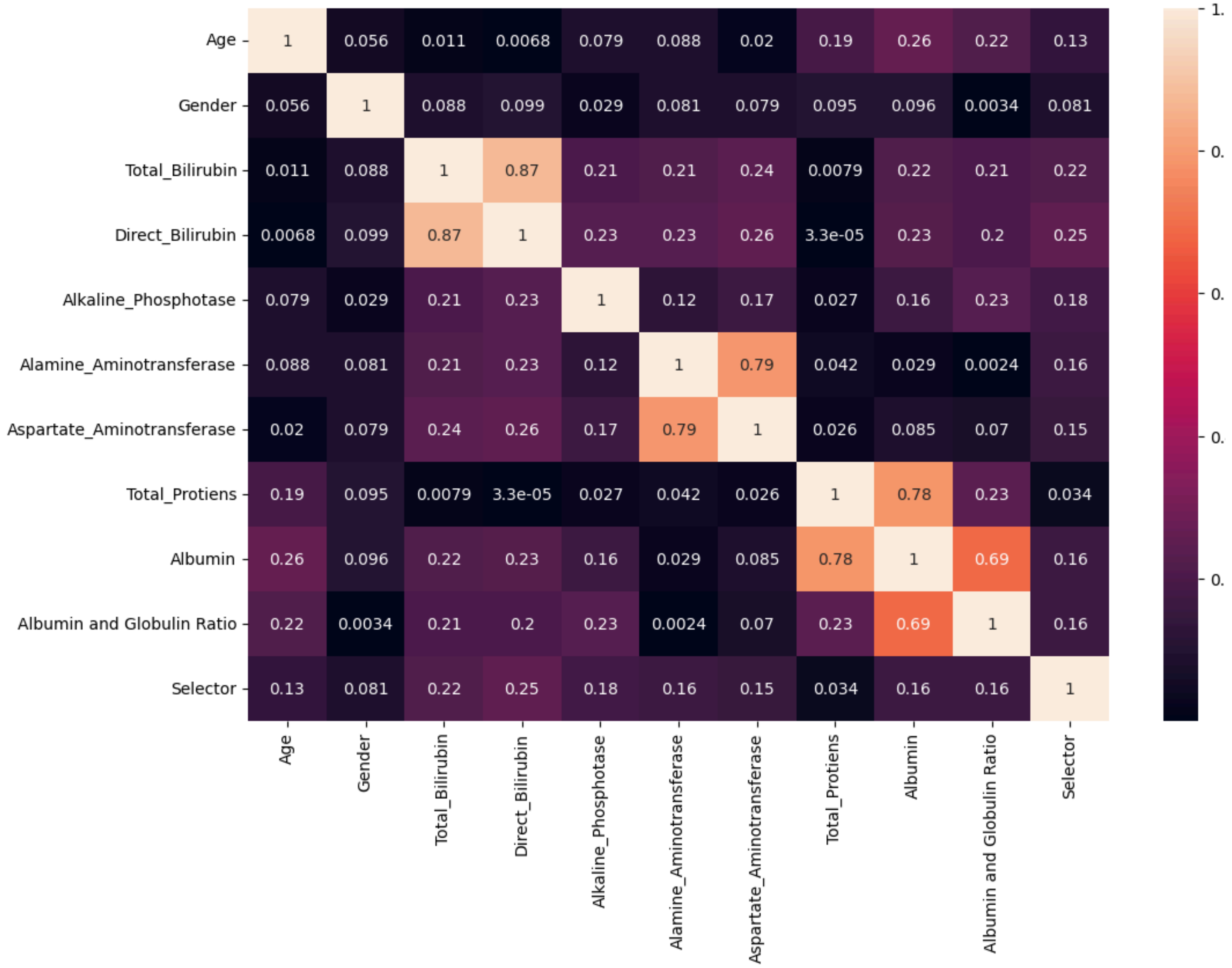
3. Gender histogram



4. Selector histogram



5. Correlation heatmap



Data Mining strategies

Data mining deals with identification of patterns in the distribution of data. From building classifiers to identifying association between different attributes, the data mining strategies can be very vast. In this task, you will use generative AI to create a classifier that will take the best 5 identified features and create a Liver Disease detection classifier. To further illustrate the power of the generative AI tools, you can include the following aspects to the code.

1. Perform standard scaling on the attributes.

2. Compare the classification accuracy of multiple classifiers on the data to identify the best one. The ones that should be used are:
   a. Logistic regression
   b. KNN
   c. Naive Bayes
   d. Decision trees
   e. Random forests
   f. Multilayer perceptron

You can create this code simply by using the following prompt.

```
1
2
3
4
5
6
7
8
9
```

```
Write a python code that can perform the following tasks:
1. Perform standard scaling operation on the top 5 attributes identified in the previous code.
2. Train and test the following classifiers on the data and print a comparative table for their accuracy performance:
   a. Logistic Regression
   b. KNN
   c. Naive Bayes
   d. Decision Trees
   e. Random Forests
   f. Multi layer perceptron
```

Copied!

You can expect a code to be generated, similar to the one shown below.

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

# Standard scaling on the top 5 attributes
scaler = StandardScaler()
top_5_attributes = data[top_5_features.index]
scaled_data = scaler.fit_transform(top_5_attributes)
scaled_data = pd.DataFrame(scaled_data, columns=top_5_attributes.columns)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(scaled_data, data['Selector'], test_size=0.2, random_state=42)

# Train and test classifiers
classifiers = {
    'Logistic Regression': LogisticRegression(),
    'KNN': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB(),
    'Decision Trees': DecisionTreeClassifier(),
    'Random Forests': RandomForestClassifier(),
    'Multi layer perceptron': MLPClassifier()
}

results = {}
for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

# Create DataFrame for the results
results_df = pd.DataFrame(list(results.items()), columns=['Classifier', 'Accuracy'])
print(results_df)
```

Copied!

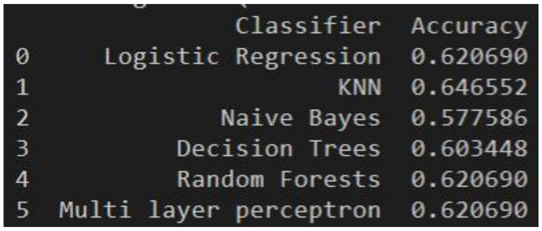
You can use the above code and modify your existing file to add the data mining aspects to it. The resulting code is expected to look as shown below.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Load the CSV data into a pandas data frame
data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/ILPD.c

# Save histograms of data distribution for 'Age', 'Gender', and 'Selector'
plt.figure()
data['Age'].hist()
plt.savefig('age_histogram.png')

plt.figure()
data['Gender'].hist()
plt.savefig('gender_histogram.png')

plt.figure()
data['Selector'].hist()
plt.savefig('selector_histogram.png')

# Save correlation heatmap of the data set
plt.figure(figsize=(12,8))
```

```
corr = data.corr()
sns.heatmap(abs(corr), annot=True)
plt.savefig('correlation_heatmap.png', bbox_inches='tight')

# Identify top 5 features with highest absolute correlation with 'Selector'
correlation_with_selector = corr['Selector'].abs().sort_values(ascending=False)
top_5_features = correlation_with_selector[1:6]  # Exclude 'Selector' itself
#print(top_5_features)

# Standard scaling on the top 5 attributes
scaler = StandardScaler()
top_5_attributes = data[top_5_features.index]
scaled_data = scaler.fit_transform(top_5_attributes)
scaled_data = pd.DataFrame(scaled_data, columns=top_5_attributes.columns)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(scaled_data, data['Selector'], test_size=0.2, random_state=42)

# Train and test classifiers
classifiers = {
    'Logistic Regression': LogisticRegression(),
    'KNN': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB(),
    'Decision Trees': DecisionTreeClassifier(),
    'Random Forests': RandomForestClassifier(),
    'Multi layer perceptron': MLPClassifier()
}

results = {}
for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

# Create DataFrame for the results
results_df = pd.DataFrame(list(results.items()), columns=['Classifier', 'Accuracy'])
print(results_df)
```

Copied!

As the final output, you can expect the following data frame to be printed.



```
               Classifier  Accuracy
0     Logistic Regression  0.620690
1                     KNN  0.646552
2             Naive Bayes  0.577586
3          Decision Trees  0.603448
4          Random Forests  0.620690
5  Multi layer perceptron  0.620690
```

Conclusion

Congratulations on completing this lab!

You now know how to use generative AI to:

1. Perform exploratory data analysis
2. Implement data mining strategies

Author(s)

Abhishek Gagneja