Assignment No 2

Aim

Perform the following operations using R/Python on the data sets:

a) Compute and display summary statistics for each feature available in the dataset. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles)

b) Illustrate the feature distributions using histogram.

c) Data cleaning, Data integration, Data transformation, Data model building (e.g.

Classification)

Objective

The objective of this assignment is to provide practical experience in performing summary statistics, visualizing feature distributions through histograms, applying data cleaning and transformation techniques, integrating datasets, and finally, constructing a basic classification model. This enhances both analytical and modeling skills using R or Python.

Theoretical

In the data science workflow, understanding the dataset is the first essential step. This is achieved through summary statistics and visualizations. Cleaning and transforming data ensures consistency and completeness. Data integration brings multiple sources into a cohesive structure. Finally, model building—especially classification—applies machine learning to predict categorical outcomes based on feature patterns.

Methods and Explanations of Operations

a) Compute and Display Summary Statistics

- - Summary statistics provide an overview of numerical columns in the dataset.
- - Includes: minimum, maximum, mean, median, range, standard deviation, variance, and percentiles (25th, 50th, 75th).
- - Why Important: Helps understand the distribution, spread, and central tendency of the data.

b) Illustrate Feature Distributions using Histogram

- - Histograms are graphical representations of the distribution of a numerical feature.
- - They divide data into bins and count how many data points fall into each bin.
- - Why Important: Helps to detect skewness, modality, and potential outliers in the data.

c) Data Cleaning, Integration, Transformation, and Model Building

- - Data Cleaning: Handling missing values, duplicates, inconsistencies, and errors.
- - Data Integration: Merging datasets from different sources into a unified dataset.
- - Data Transformation: Standardizing formats, normalizing numerical values, and encoding categorical variables.
- - Model Building (Classification): Applying supervised learning to predict categories (e.g. logistic regression, decision tree).
- - Why Important: These processes ensure high-quality input for modeling and enable meaningful predictions.

Outputs:

jupyter ML_Assignment_2 Last Checkpoint: last month

File Edit View Run Kernel Settings Help Not Trusted

Code JupyterLab Python 3 (ipykernel)

[73]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

[74]:
```
df = pd.read_csv('/content/Customers.csv')
```

[75]:
```
df.head()
```

[75]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6

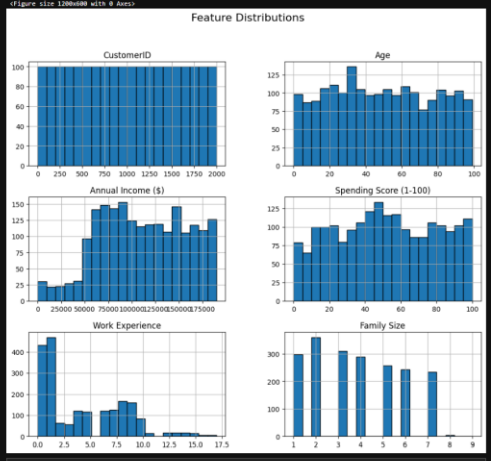
[76]:
```
print("Summary Statistics:")
print(df.describe(include='all'))
```

```
Summary Statistics:
        CustomerID  Gender          Age  Annual Income ($)  \
count  2000.000000    2000  2000.000000        2000.000000
unique         NaN       2          NaN                NaN
top            NaN  Female          NaN                NaN
freq           NaN    1186          NaN                NaN
mean    1000.500000     NaN    48.960000      110731.821500
std      577.494589     NaN    28.429747       45739.536688
min        1.000000     NaN     0.000000           0.000000
25%      500.750000     NaN    25.000000       74572.000000
50%     1000.500000     NaN    48.000000      110047.000000
75%     1500.250000     NaN    73.000000      149092.750000
max     2000.000000     NaN    99.000000      189974.000000

       Spending Score (1-100) Profession  Work Experience  Family Size
count             2000.000000       1965      2000.000000  2000.000000
unique                    NaN          9              NaN          NaN
top                       NaN     Artist              NaN          NaN
freq                      NaN        612              NaN          NaN
mean                50.962500        NaN         4.182500     3.768500
std                 27.934661        NaN         3.922204     1.978749
min                  0.000000        NaN         0.000000     1.000000
25%                 28.000000        NaN         1.000000     2.000000
50%                 50.000000        NaN         3.000000     4.000000
75%                 75.000000        NaN         7.000000     5.000000
max                100.000000        NaN        17.000000     9.000000
```

[77]:
```
plt.figure(figsize=(12,8))
df.hist(figsize=(12,10), bins=20, edgecolor='black')
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



[78]:
```
encoder = LabelEncoder()
df['Gender'] = encoder.fit_transform(df['Gender'])
df['Profession'] = encoder.fit_transform(df['Profession'])
```

[79]:
```
df.fillna(df.median(), inplace=True)
```

[80]:
```
scaler = StandardScaler()
df[['Annual Income ($)', 'Spending Score (1-100)', 'Work Experience', 'Family Size']] = scaler.fit_transform(
    df[['Annual Income ($)', 'Spending Score (1-100)', 'Work Experience', 'Family Size']])
```

[81]:
```
df['Spending Category'] = pd.cut(df['Spending Score (1-100)'], bins=[0, 33, 66, 100], labels=["Low", "Medium", "High"])
df['Spending Category'] = encoder.fit_transform(df['Spending Category'])
```

[82]:
```
X = df.drop(columns=['CustomerID', 'Spending Score (1-100)', 'Spending Category'])
y = df['Spending Category']
```

[83]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[84]:
```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

[84]:
```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

RandomForestClassifier
?Documentation for RandomForestClassifier iFitted
RandomForestClassifier(random_state=42)

[85]:
```
y_pred = model.predict(X_test)
```

[86]:
```
print("Classification Report:")
print(classification_report(y_test, y_pred))
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
```

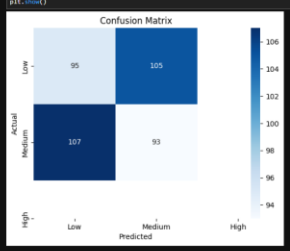
```
Classification Report:
              precision    recall  f1-score   support

           0       0.47      0.47      0.47       200
           1       0.47      0.47      0.47       200

    accuracy                           0.47       400
   macro avg       0.47      0.47      0.47       400
weighted avg       0.47      0.47      0.47       400

Accuracy Score: 0.47
```

[87]:
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Low', 'Medium', 'High'], yticklabels=['Low', 'Medium', 'High'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



Conclusion

We have successfully performed descriptive statistics, histogram-based visualizations, data cleaning, integration, and transformation followed by basic classification modeling using R/Python. This assignment has provided a comprehensive understanding of data preparation and how it feeds into effective model building. Through this assignment, I learned how each preprocessing step influences the accuracy and reliability of data analysis and predictions.