Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)

Degree & Branch	B.E. Computer Science & Engineering	Semester	V
Subject Code & Name	ICS1512 & Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Batch:2023-2028	**Due date: 27.01.2025**

Experiment 1: Working with Python packages-Numpy, Scipy, Scikit-Learn, Matplotlib

Aim: To study and explore the fundamental Python libraries used in data science and machine learning, and to understand how different datasets can be analyzed and mapped to appropriate machine learning models using exploratory data analysis techniques.

Libraries Used:

- **NumPy**: Used for numerical computations and efficient handling of multi-dimensional arrays.
- **Pandas**: Used for data manipulation, cleaning, and analysis using DataFrames.
- **Matplotlib**: Used for creating visualizations such as line graphs, bar charts, and histograms.
- **Seaborn**: Used for advanced statistical data visualization with attractive and informative plots.

Objectives performed:

1 Iris Dataset

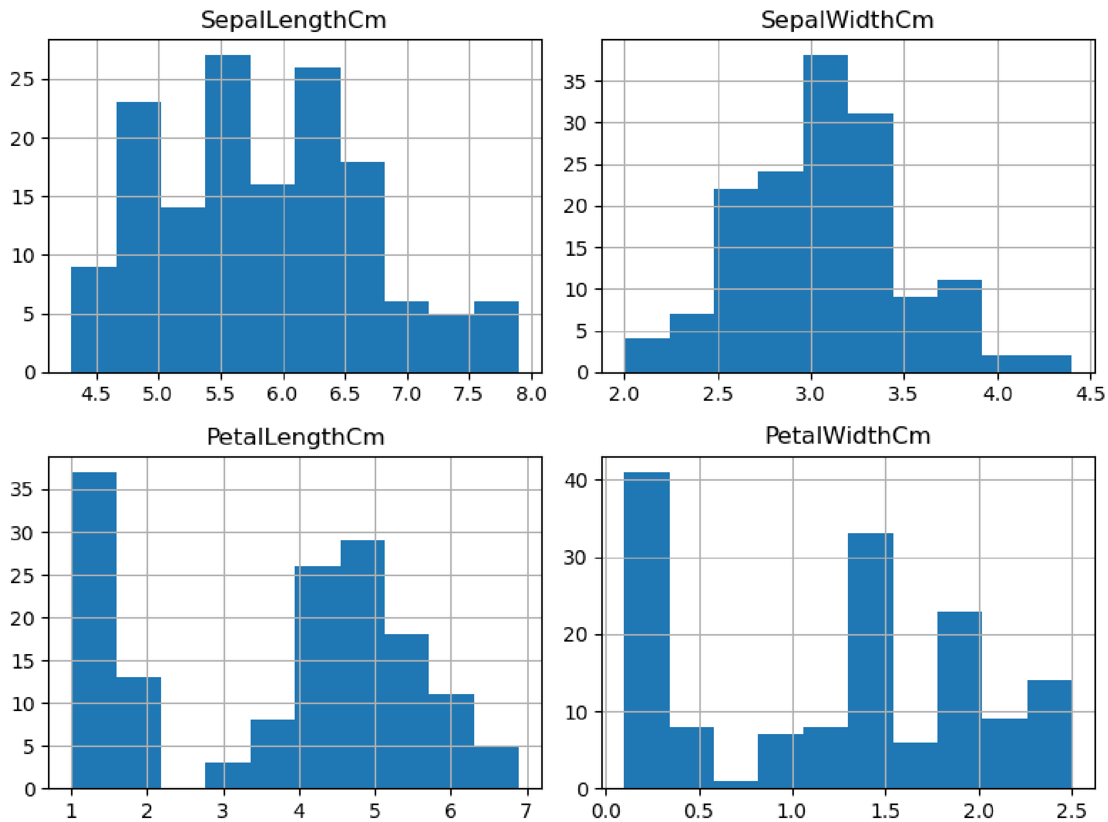
[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder

from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import f_classif
from sklearn.model_selection import train_test_split
```

[5]:
```
df=pd.read_csv('Datasets/Iris.csv')
df.drop(columns=['Id'], inplace=True)

df.head()
```

[6]:
```
df.info()
```

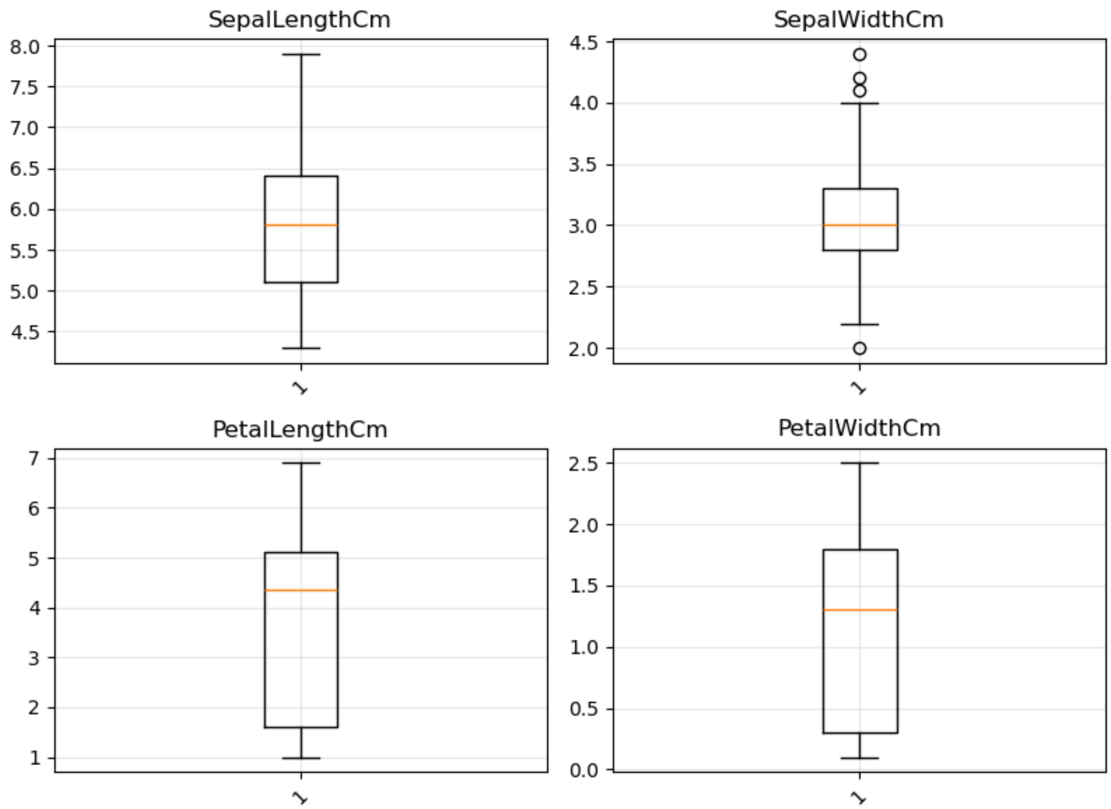
[7]:
```
df.describe()
```

[8]:
```
df[df.columns].hist(figsize=(8,6))
plt.tight_layout()
plt.savefig("histogram.png", dpi=300, bbox_inches="tight")
plt.show()
```

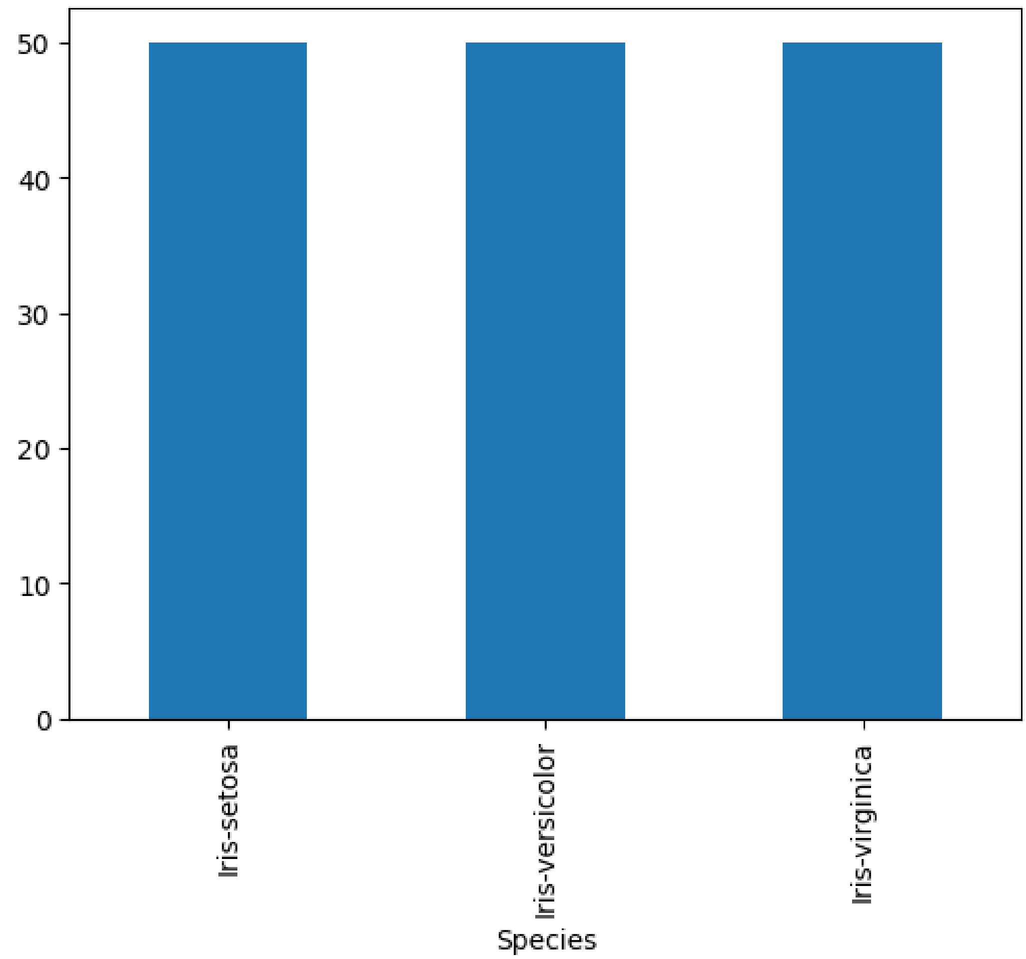


[9]:
```
num_cols = df.select_dtypes(include=['int64','float64']).columns

fig, axes = plt.subplots(2,2, figsize = (8,6))
axes = axes.flatten()

for i, col in enumerate(num_cols):
  axes[i].boxplot(df[col])
  axes[i].set_title(col)
  axes[i].tick_params(axis='x', rotation=45)
  axes[i].grid(True, alpha=0.3)

plt.tight_layout()
```

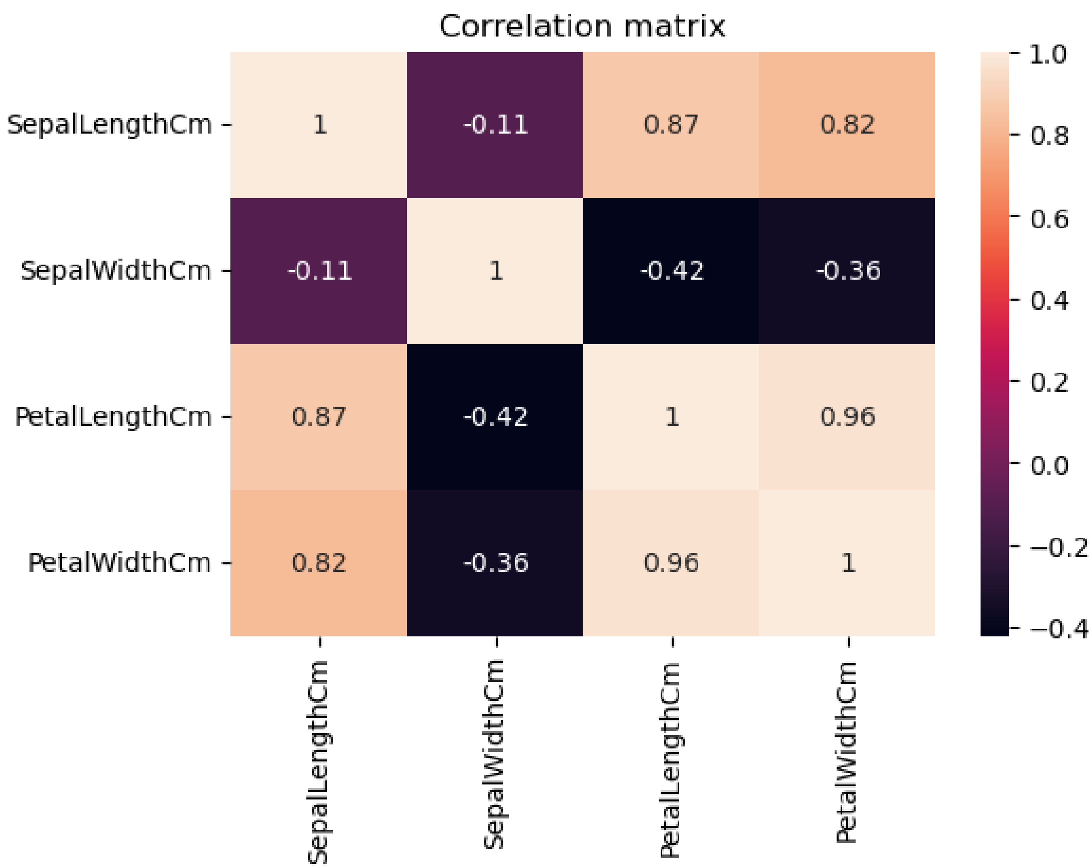
```
plt.savefig("box_plot.png", dpi=300, bbox_inches="tight")
plt.show()
```



[10]:
```
df['Species'].value_counts().plot(kind='bar')
axes[i].set_title("Species")
axes[i].set_xlabel('')
axes[i].set_ylabel('Count')
plt.savefig("bar_plot.png", dpi=300, bbox_inches="tight")
```



[11]:
```
plt.figure(figsize=(6,4))
sns.heatmap(df[num_cols].corr(), annot=True)
plt.title("Correlation matrix")
plt.savefig("correlational_matrix.png", dpi=300, bbox_inches="tight")
plt.show()
```




```
[ ]:
```

2 Loan Amount Prediction

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import scipy

     from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score, confusion_matrix,␣
      ↪classification_report
     from sklearn.impute import SimpleImputer
```

```
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import f_classif
from sklearn.model_selection import train_test_split
```

[ ]:
```
#Loan dataset
df = pd.read_csv("/content/loan_data.csv")
print(df)
```

Shape

[ ]:
```
print("Describe")
df.describe()
```

Describe

[ ]:
```
print("columns")
df.columns
```

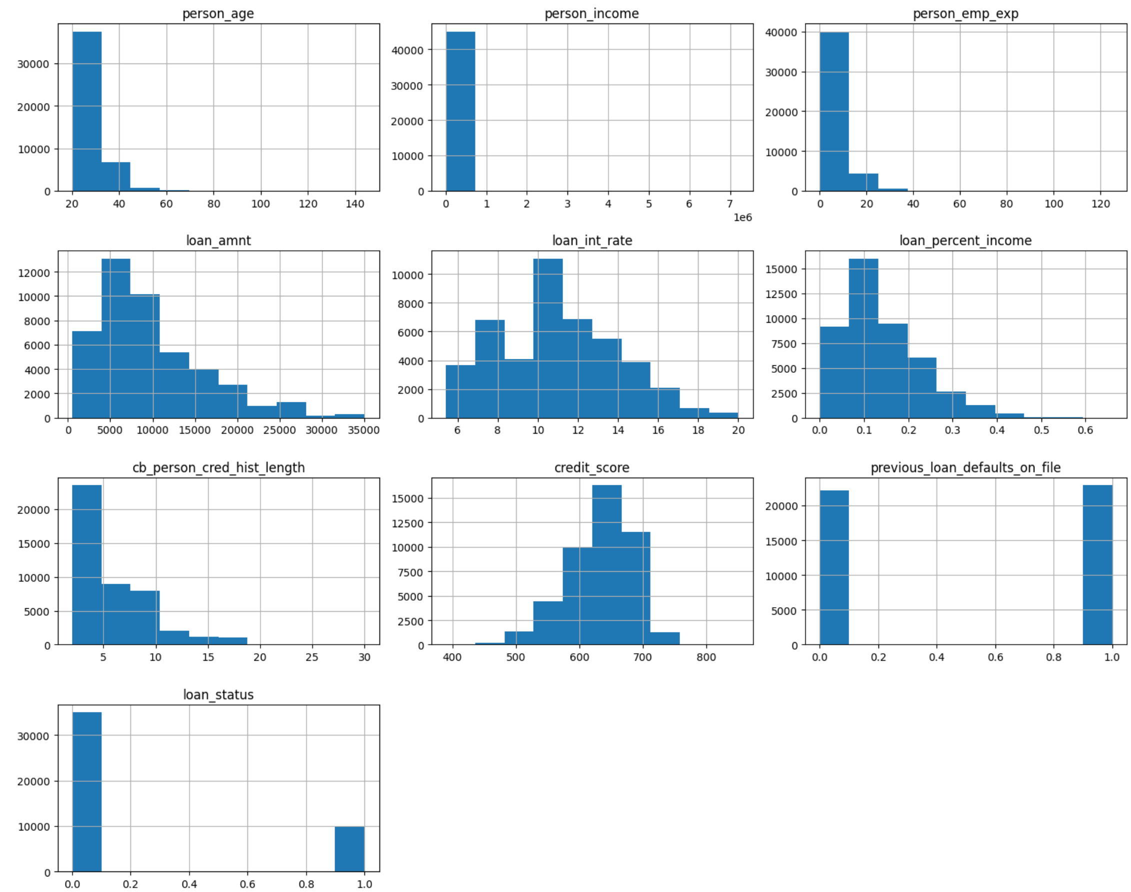
columns

[ ]:
```
df.isnull().sum()
(df.isnull().mean() * 100).sort_values(ascending=False)
```

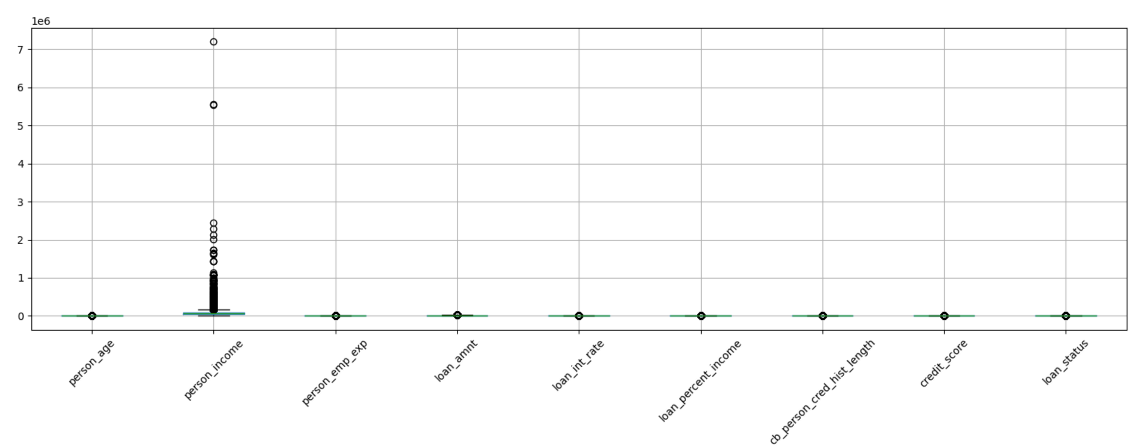
[ ]:
```
#including previous_loan_defaults_on_file
df['previous_loan_defaults_on_file'] = df['previous_loan_defaults_on_file'].
↪map({
    'Yes': 1,
    'No': 0
})

#column selection

num_cols = df.select_dtypes(include=['int64','float64']).columns
cat_cols = df.select_dtypes(include=['object']).columns

num_cols, cat_cols
```

[ ]:
```
df[num_cols].describe()
#df[num_cols].median()
```

[ ]:
```
df[num_cols].hist(figsize=(15,12))
plt.tight_layout()
plt.show()
```

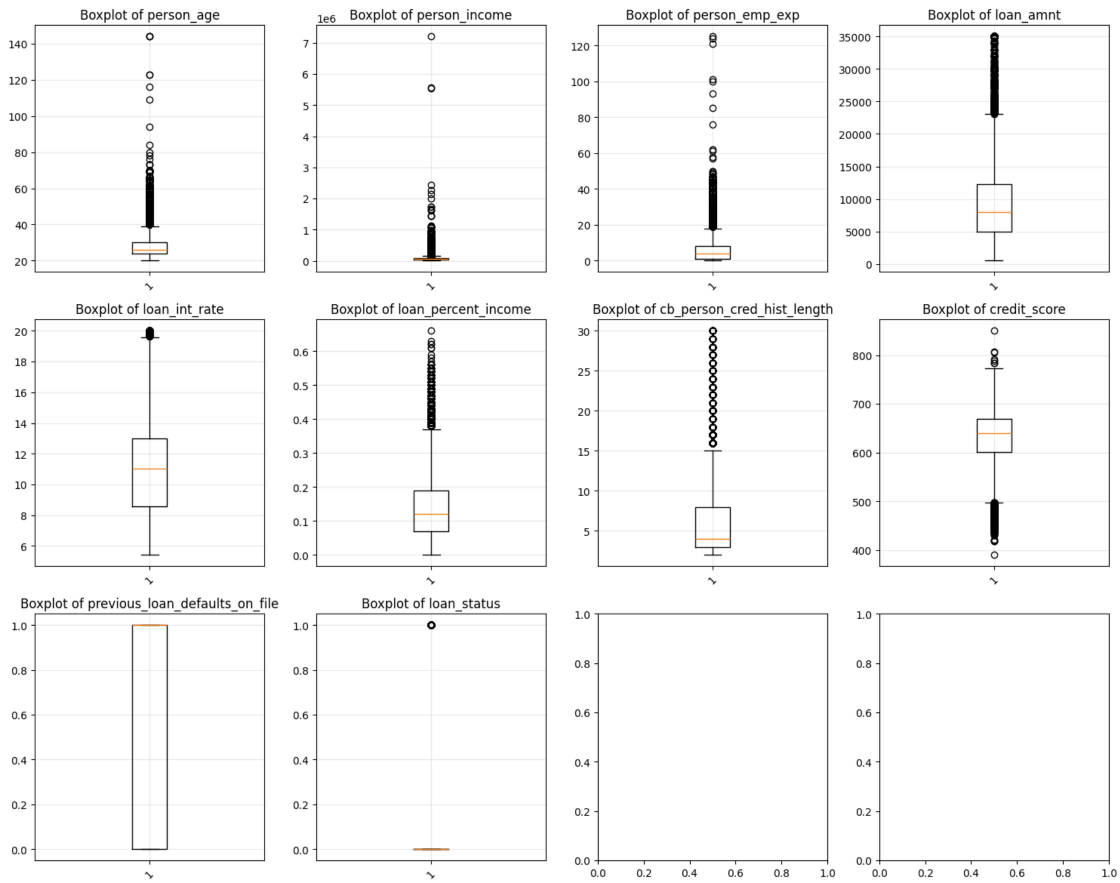


[ ]:
```
#box plot
plt.figure(figsize=(15, 6))
df[num_cols].boxplot(rot=45)
plt.tight_layout()
plt.show()
```



[ ]:

```
fig, axes = plt.subplots(3, 4, figsize=(15, 12))
axes = axes.flatten()

for i, col in enumerate(num_cols):
    axes[i].boxplot(df[col])
    axes[i].set_title(f"Boxplot of {col}")
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



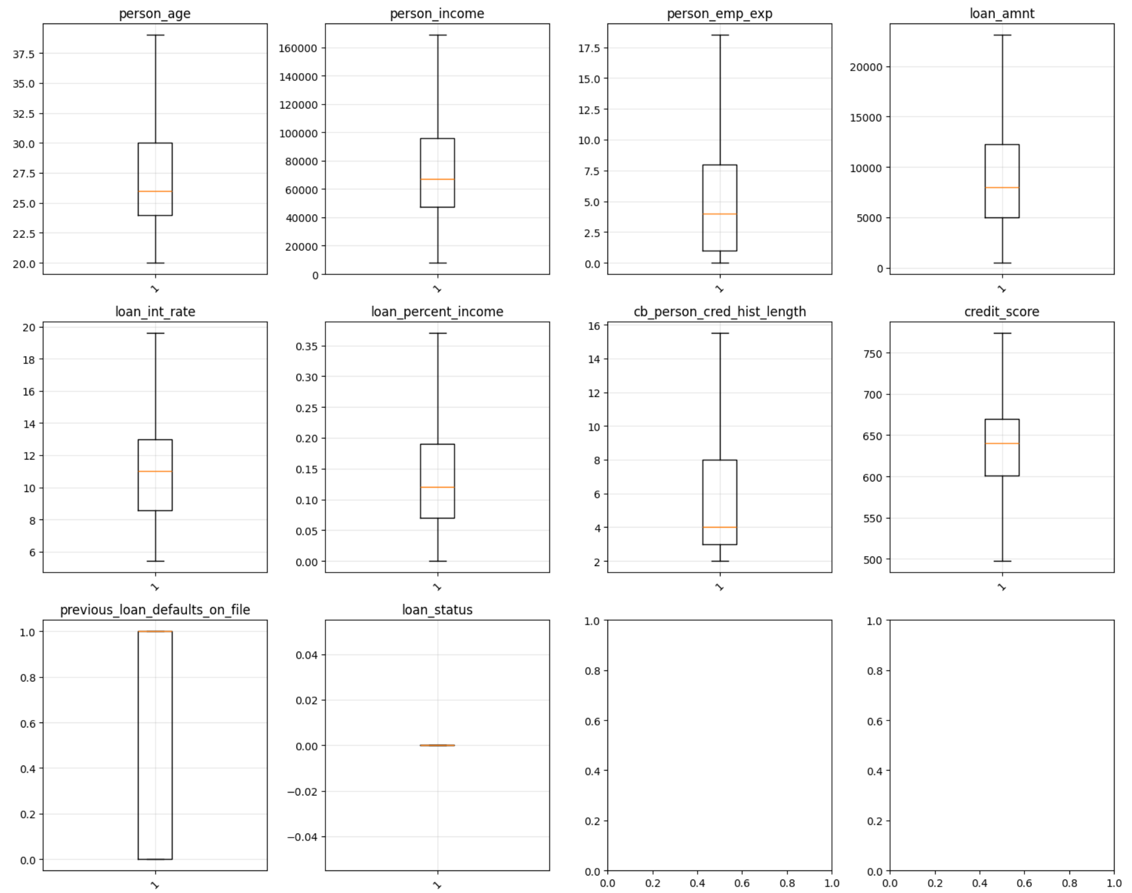
[ ]:

```
outlier_summary = {}

for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
```

```
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    lower_outliers = (df[col] < lower_bound).sum()
    upper_outliers = (df[col] > upper_bound).sum()

    outlier_summary[col] = {
        "Lower Outliers": lower_outliers,
        "Upper Outliers": upper_outliers,
        "Total Outliers": lower_outliers + upper_outliers
    }

outlier_df = pd.DataFrame(outlier_summary).T
outlier_df
```

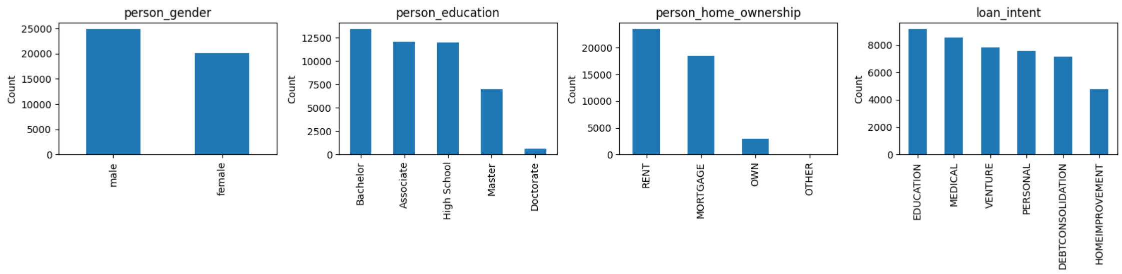
[ ]:
```
#winorization
df_capped = df.copy()

for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df_capped[col] = df_capped[col].clip(lower_bound, upper_bound)
```

[ ]:
```
fig, axes = plt.subplots(3, 4, figsize=(15, 12))
axes = axes.flatten()

for i, col in enumerate(num_cols):
    axes[i].boxplot(df_capped[col])
    axes[i].set_title(col)
    axes[i].tick_params(axis='x', rotation=45)
    axes[i].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

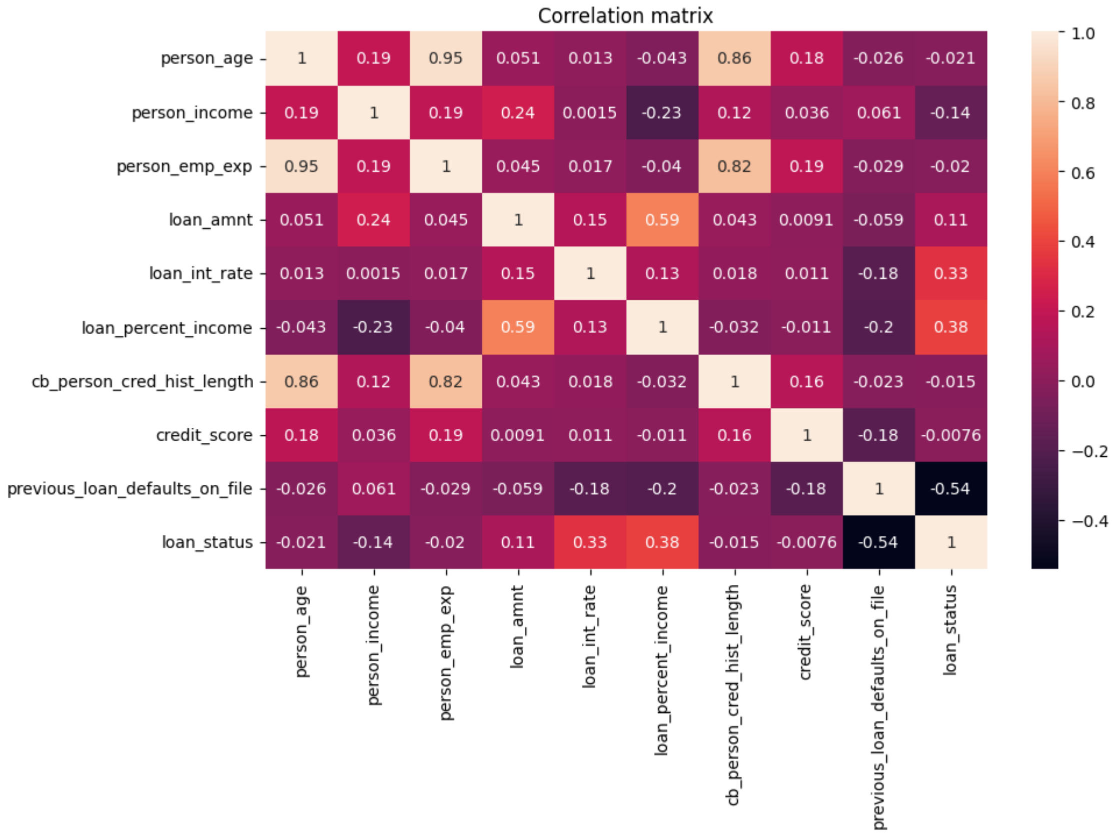


[ ]:
```
ncols = 5
nrows = (len(cat_cols) + ncols - 1) // ncols

fig, axes = plt.subplots(
    nrows=nrows,
    ncols=ncols,
    figsize=(20, 4 * nrows)
)

# Make axes always a flat array
if isinstance(axes, np.ndarray):
    axes = axes.flatten()
else:
    axes = [axes]

# Plot each categorical column
for i, col in enumerate(cat_cols):
    df[col].value_counts().plot(kind='bar', ax=axes[i])
    axes[i].set_title(col)
```

```
    axes[i].set_xlabel('')
    axes[i].set_ylabel('Count')

# Remove any unused subplots
for j in range(len(cat_cols), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



[ ]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df[num_cols].corr(), annot=True)
plt.title("Correlation matrix")
plt.show()
```




```
[ ]: y = df['loan_status']
     X = df.drop(columns=['loan_status'])

     num_cols = [col for col in num_cols if col != 'loan_status']
```

```
[ ]: # ANOVA

     selector = SelectKBest(score_func=f_classif, k=5)
     X_anova_selected = selector.fit_transform(X[num_cols], y)

     anova_scores = pd.DataFrame({
         'Feature': num_cols,
         'ANOVA F-Score': selector.scores_
     }).sort_values(by='ANOVA F-Score', ascending=False)

     anova_scores
```

```
[ ]:                          Feature  ANOVA F-Score
     8  previous_loan_defaults_on_file   18824.727466
     5             loan_percent_income    7824.794030
```

```
4                  loan_int_rate    5574.454260
1                  person_income     845.525887
3                      loan_amnt     528.213632
0                     person_age      20.763596
2                 person_emp_exp      18.883771
6     cb_person_cred_hist_length       9.926174
7                   credit_score       2.631606
```

[ ]:
```python
selected_features = [
    'previous_loan_defaults_on_file',
    'loan_percent_income',
    'loan_int_rate',
    'person_income',
    'loan_amnt'
]
```

[ ]:
```python
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(
    X,
    y,
    test_size=0.30,         # 30% → temp
    random_state=42,
    stratify=y              # IMPORTANT for imbalanced classes
)


X_val, X_test, y_val, y_test = train_test_split(
    X_temp,
    y_temp,
    test_size=0.50,         # split 30% into 15% + 15%
    random_state=42,
    stratify=y_temp
)

print("Training set:", X_train.shape)
print("Validation set:", X_val.shape)
print("Test set:", X_test.shape)
```

[ ]:
```python
#to verify class imbalance
def class_distribution(y, name):
    print(f"{name} class distribution:")
    print(y.value_counts(normalize=True))
    print()

class_distribution(y_train, "Train")
class_distribution(y_val, "Validation")
```

```
class_distribution(y_test, "Test")
```

3 Predicting Diabetes

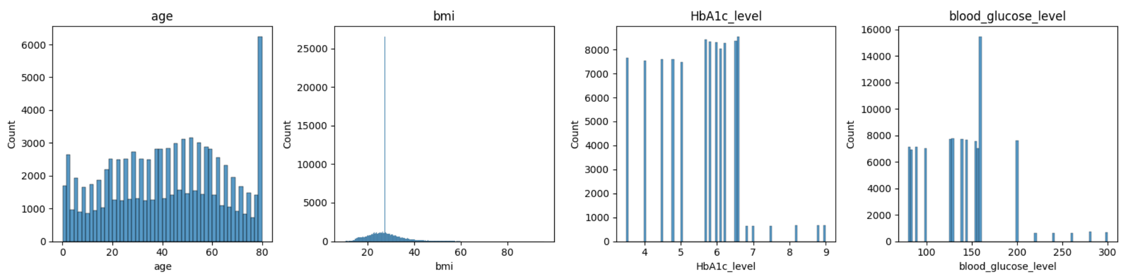
[ ]:
```
# DiabetesData
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder

from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import f_classif
from sklearn.model_selection import train_test_split
```

[ ]:
```
df=pd.read_csv('diabetes.csv')
df
```

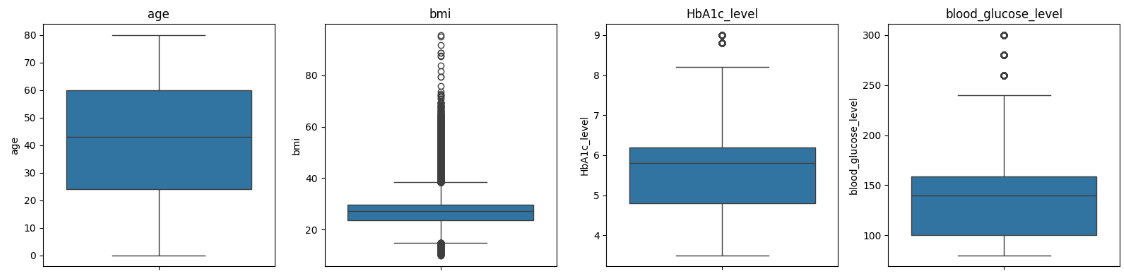
[ ]:
```
print(df.columns)
```

[ ]:
```
print(df.describe())
```

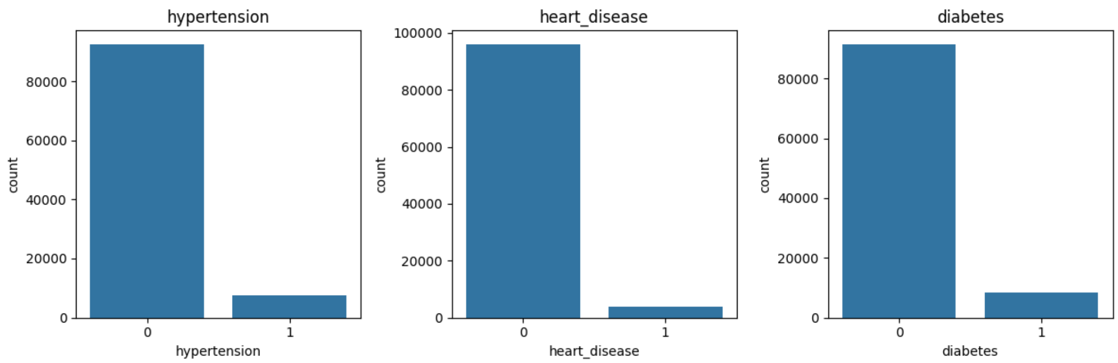
[ ]:
```
import math
num_cols = ['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']
ncols=5
nrows=math.ceil(len(num_cols)/ncols)
fig,axes=plt.subplots(nrows=nrows,ncols=ncols,figsize=(20,4*nrows))
axes=axes.flatten()
for i,col in enumerate(num_cols):
    sns.histplot(df[col],ax=axes[i])
    axes[i].set_title(col)
for j in range(i+1,len(axes)):
    fig.delaxes(axes[j])
plt.tight_layout()
plt.show()
```



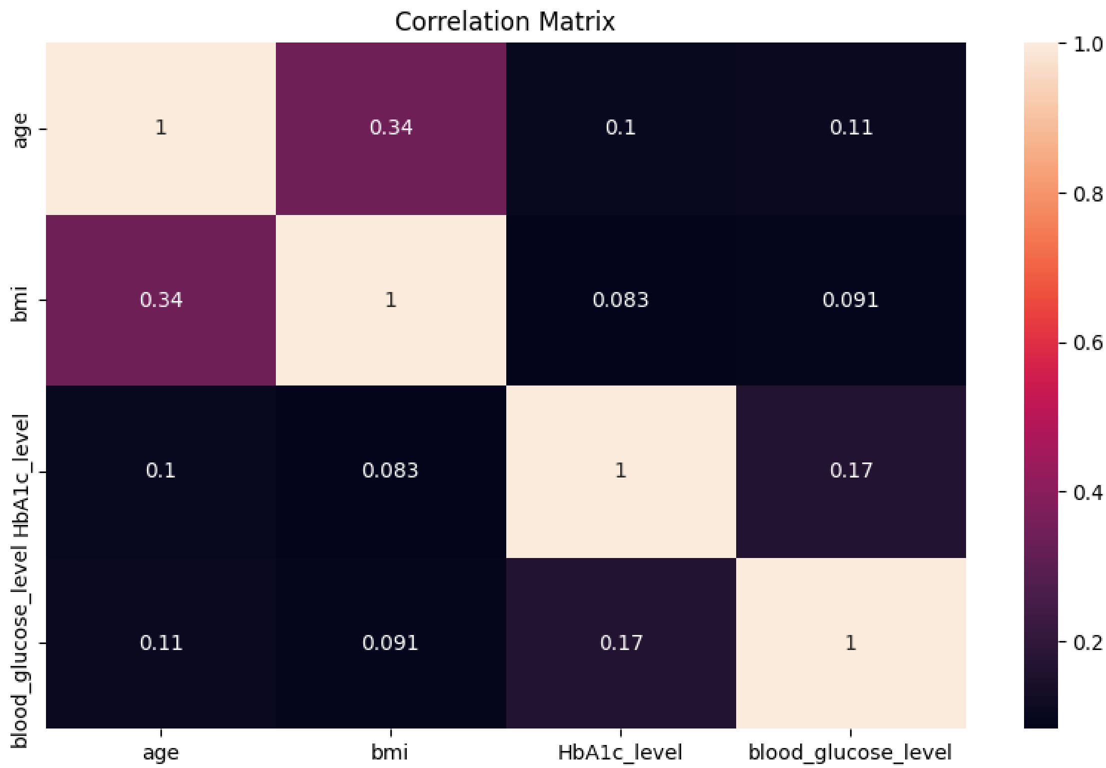
[ ]:
```
ncols=5
nrows=math.ceil(len(num_cols)/ncols)
fig,axes=plt.subplots(nrows=nrows,ncols=ncols,figsize=(20,4*nrows))
axes=axes.flatten()
for i,col in enumerate(num_cols):
    sns.boxplot(df[col],ax=axes[i])
    axes[i].set_title(col)
for j in range(i+1,len(axes)):
    fig.delaxes(axes[j])
plt.tight_layout()
plt.show()
```



[ ]:
```
cat_cols = ['hypertension', 'heart_disease', 'diabetes']

fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(20,4))
axes = axes.flatten()
for i, col in enumerate(cat_cols):
    sns.countplot(x=df[col], ax=axes[i])
    axes[i].set_title(col)


for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



[ ]:
```
plt.figure(figsize=(10, 6))
sns.heatmap(df[num_cols].corr(), annot=True)
plt.title("Correlation Matrix")
plt.show()
```



4 Classification of Email Spam

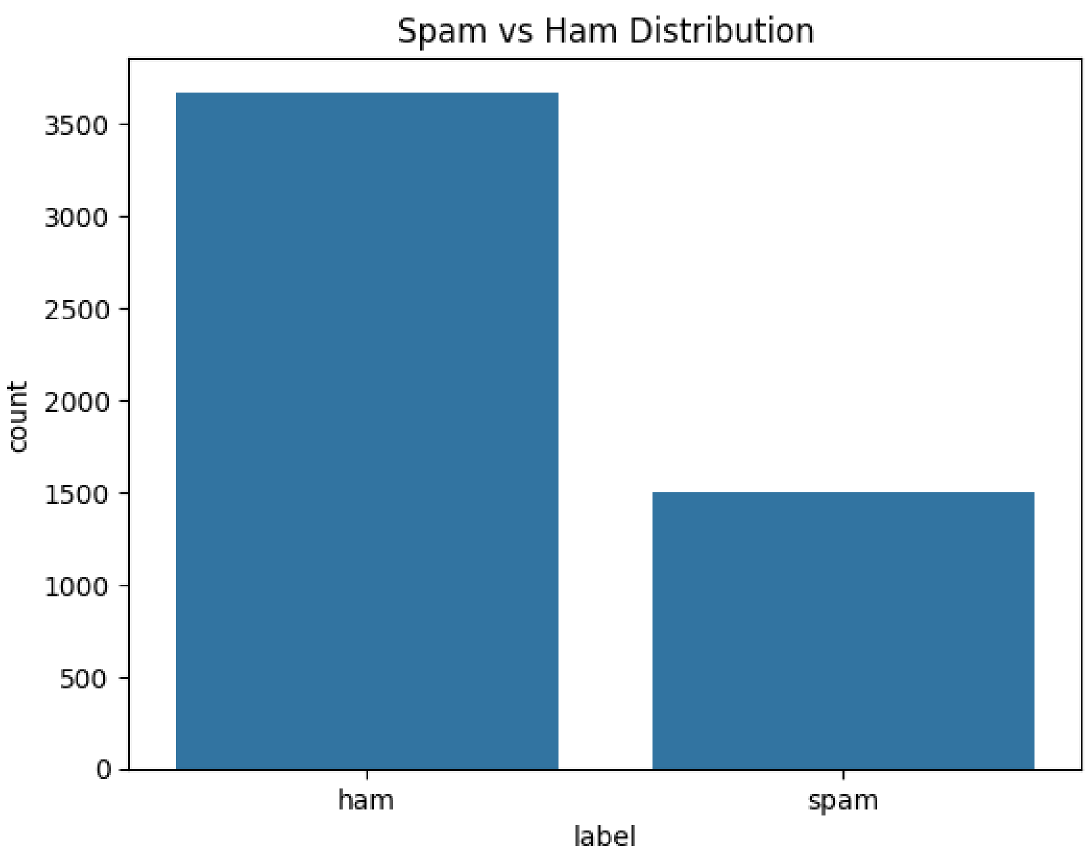
[19]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import math
from sklearn.feature_extraction.text import TfidfVectorizer
```

[20]:
```python
df = pd.read_csv('spam_ham_dataset.csv')
df
```

[21]:
```python
df.info()
```

[22]:
```python
df.describe()
```

[23]:
```python
sns.countplot(x='label', data=df)
plt.title('Spam vs Ham Distribution')
plt.show()
```



[24]:
```
import re
def clean_text(text):
  text = text.lower()
  text = re.sub(r"http\S+", "", text)
  text = re.sub(r"[^a-z\s]", "", text)
  return text

df['clean_text'] = df['text'].apply(clean_text)
df['word_count'] = df['clean_text'].apply(lambda x: len(x.split()))
df['char_count'] = df['clean_text'].apply(len)

df.head()
```

[25]:
```
from collections import Counter

all_words = " ".join(df['clean_text']).split()
common_words = Counter(all_words).most_common(20)
common_words
```

[26]:
```
spam_words = " ".join(df[df['label']=="spam"]['clean_text']).split()
ham_words = " ".join(df[df['label']=="ham"]['clean_text']).split()
print("Spam words most common \n\n", Counter(spam_words).most_common(15))
print("\nNot Spam common words \n", Counter(ham_words).most_common(15))
```

[27]:
```
df.columns
```

[27]:
```
Index(['Unnamed: 0', 'label', 'text', 'label_num', 'clean_text', 'word_count',
       'char_count'],
      dtype='object')
```

[28]:
```
df.drop(columns=['Unnamed: 0'], inplace=True)
```

[29]:
```
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=5000
)

X_text = vectorizer.fit_transform(df['clean_text'])
y = df['label_num']
```

[32]:
```
from sklearn.feature_selection import SelectKBest, chi2

chi2_scores, p_values = chi2(X_text, y)
```

[33]:
```
chi2_df = pd.DataFrame({
    'word': vectorizer.get_feature_names_out(),
    'chi2_score': chi2_scores,
```

```
    'p_value': p_values
})

chi2_df = chi2_df.sort_values(by='chi2_score', ascending=False)

chi2_df.head(15)
```

5 Handwritten Character Recognition / MNIST

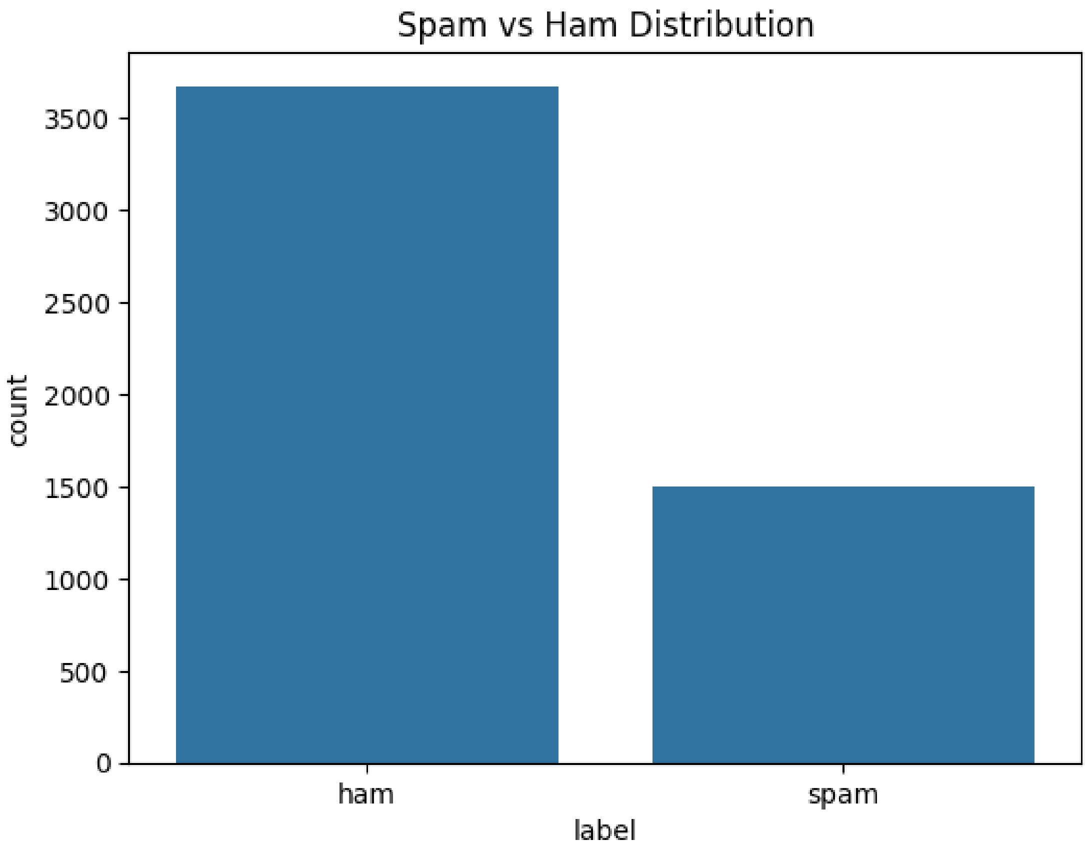
[19]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import math
from sklearn.feature_extraction.text import TfidfVectorizer
```

[20]:
```
df = pd.read_csv('spam_ham_dataset.csv')
df
```

[21]:
```
df.info()
```

[22]:
```
df.describe()
```

[23]:
```
sns.countplot(x='label', data=df)
plt.title('Spam vs Ham Distribution')
plt.show()
```



[24]:
```
import re
def clean_text(text):
  text = text.lower()
  text = re.sub(r"http\S+", "", text)
  text = re.sub(r"[^a-z\s]", "", text)
  return text

df['clean_text'] = df['text'].apply(clean_text)
df['word_count'] = df['clean_text'].apply(lambda x: len(x.split()))
df['char_count'] = df['clean_text'].apply(len)

df.head()
```

[25]:
```
from collections import Counter

all_words = " ".join(df['clean_text']).split()
common_words = Counter(all_words).most_common(20)
common_words
```

[26]:
```
spam_words = " ".join(df[df['label']=="spam"]['clean_text']).split()
ham_words = " ".join(df[df['label']=="ham"]['clean_text']).split()
print("Spam words most common \n\n", Counter(spam_words).most_common(15))
print("\nNot Spam common words \n", Counter(ham_words).most_common(15))
```

[27]:
```
df.columns
```

[27]:
```
Index(['Unnamed: 0', 'label', 'text', 'label_num', 'clean_text', 'word_count',
       'char_count'],
      dtype='object')
```

[28]:
```
df.drop(columns=['Unnamed: 0'], inplace=True)
```

[29]:
```
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=5000
)

X_text = vectorizer.fit_transform(df['clean_text'])
y = df['label_num']
```

[32]:
```
from sklearn.feature_selection import SelectKBest, chi2

chi2_scores, p_values = chi2(X_text, y)
```

[33]:
```
chi2_df = pd.DataFrame({
    'word': vectorizer.get_feature_names_out(),
    'chi2_score': chi2_scores,
    'p_value': p_values
})

chi2_df = chi2_df.sort_values(by='chi2_score', ascending=False)

chi2_df.head(15)
```

Results and Discussions

Learning Practices

- Understand the fundamentals of data analysis using Python libraries such as NumPy, Pandas, Matplotlib, and Seaborn.
- Analyze datasets using descriptive statistics and exploratory data analysis techniques.

Dataset	Type of ML Task	Feature Selection Technique	Suitable ML Algorithm
Iris Dataset	Classification	SelectKBest	Logistic Regression
Loan Amount Prediction	Regression	Correlation Coefficient	Linear Regression
Predicting Diabetes	Classification	SelectKBest	Ridge Regression
Classification of Email Spam	Classification	Correlation Coefficient	Naïve Bayes
Handwritten Character Recognition (MNIST)	Classification	PCA	CNN

- Visualize data distributions and relationships using appropriate plots and charts..
- Identify suitable machine learning tasks such as classification and regression for different datasets.