**ASSIGNMENT-3**

**COURSE ID: CS564**

# Foundations of Machine Learning

**Submitted by:**

**Ankit Anand**

**Roll No: 2311MC04**

**Design a predictive regression model that forecasts sales based on the "Advertising.csv" dataset. Afterwards, employ logistic regression and Support Vector Machines (SVM) to predict defaulters using the "Credit.csv" and "Credit-Modified.csv" datasets. Perform a 70-30 train-test split for model evaluation and measurement of performance. Create a scatter plot with a clear separation line to visualize the data distribution. Generate a table that assesses the significance of the dataset features using the Anova test and test the significance of the derived model parameters**.

**Introduction:**

This assignment aims to construct predictive regression models and perform classification using machine learning techniques on specific datasets. The tasks involve creating a sales forecasting model based on the "Advertising.csv" dataset using regression methods and employing logistic regression and Support Vector Machines (SVM) for default prediction on the "Credit.csv" and "Credit-Modified.csv" datasets.

**Dataset Overview:**

The datasets— "Advertising.csv," "Credit.csv," and "Credit-Modified.csv"—encompass various features and target variables. Preliminary exploration and potential preprocessing steps, such as handling missing values or categorical data encoding, may be necessary.

**Predictive Regression Model for Sales Forecasting:**

For the "Advertising.csv" dataset:
1. Data exploration and Correlation Analysis: Understanding the relationships between features and sales using theoretical concepts such as linear regression assumptions (e.g., linearity, homoscedasticity).
2. Train-Test Split: Implementing a 70-30 split for model evaluation while emphasizing the significance of unbiased evaluation and avoiding overfitting.

3. Regression Techniques: Employing methods like linear regression or random forests, emphasizing the concept of ensemble learning and feature importance analysis.
4. Model Performance Evaluation: Assessing model performance using standard regression metrics (e.g., RMSE, MAE), ensuring interpretations beyond metrics.

**Logistic Regression and SVM for Default Prediction:**

Utilizing "Credit.csv" and "Credit-Modified.csv":
1. Data Preprocessing: Incorporating theoretical concepts such as scaling and addressing class imbalance to enhance model performance.
2. Train-Test Split: Emphasizing the significance of data partitioning for unbiased evaluation.
3. Classification Algorithms: Implementing logistic regression and SVM, focusing on understanding the sigmoid function in logistic regression and the margin optimization in SVM.
4. Visualization and Model Evaluation: Creating scatter plots with separation lines to visualize data distribution and model predictions. Evaluation using classification metrics (e.g., accuracy, precision, recall, F1-score).

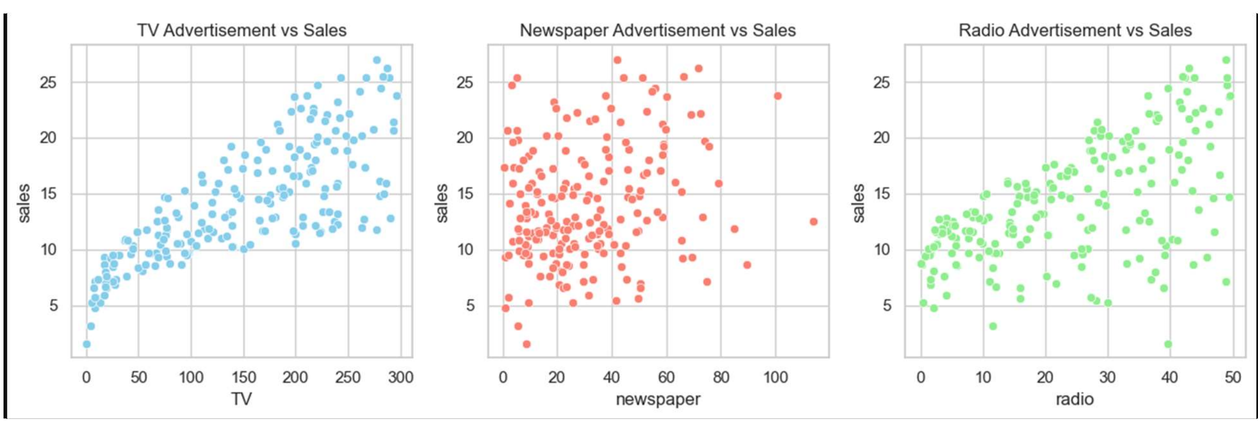
**Assessment of Feature Significance:**

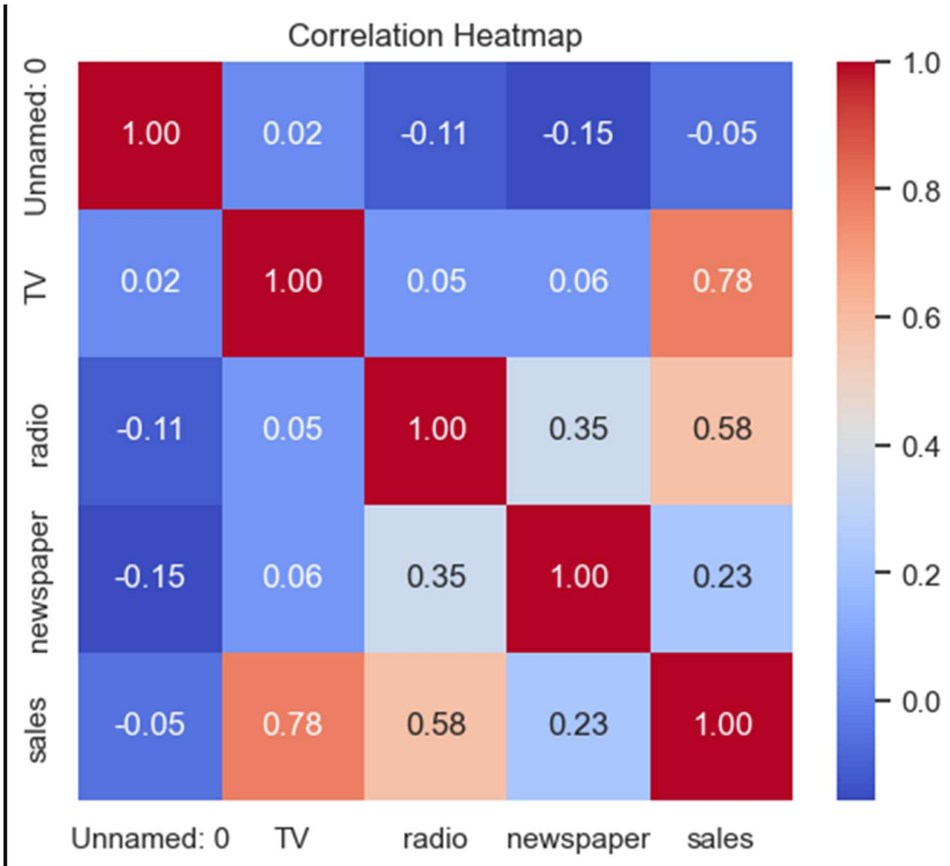
For the predictive regression model:
1. Anova Tests: Theoretical foundation of Anova tests to evaluate the significance of dataset features concerning sales forecasting.
2. Model Parameter Significance: Discussing the significance of model parameters (coefficients) through hypothesis testing or confidence intervals, ensuring the understanding of parameter interpretation.

## Sample Code with Output:

```python
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.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import statsmodels.api as sm

# Load the advertising data
advertising_data = pd.read_csv('F:/IIT 1st Semester/Assignment/ML/3/Advertising.csv')

# Visualize relationships between advertising mediums and sales
sns.set(style='whitegrid')
plt.figure(figsize=(12, 4))

plt.subplot(131)
sns.scatterplot(data=advertising_data, x='TV', y='sales', color='skyblue')
plt.title('TV Advertisement vs Sales')

plt.subplot(132)
sns.scatterplot(data=advertising_data, x='newspaper', y='sales', color='salmon')
plt.title('Newspaper Advertisement vs Sales')

plt.subplot(133)
sns.scatterplot(data=advertising_data, x='radio', y='sales', color='lightgreen')
plt.title('Radio Advertisement vs Sales')

plt.tight_layout()
plt.show()

# Heatmap to visualize correlations
plt.figure(figsize=(6, 5))
sns.heatmap(advertising_data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

# Preprocess the data
X = advertising_data.drop('sales', axis=1)
y = advertising_data['sales']
```

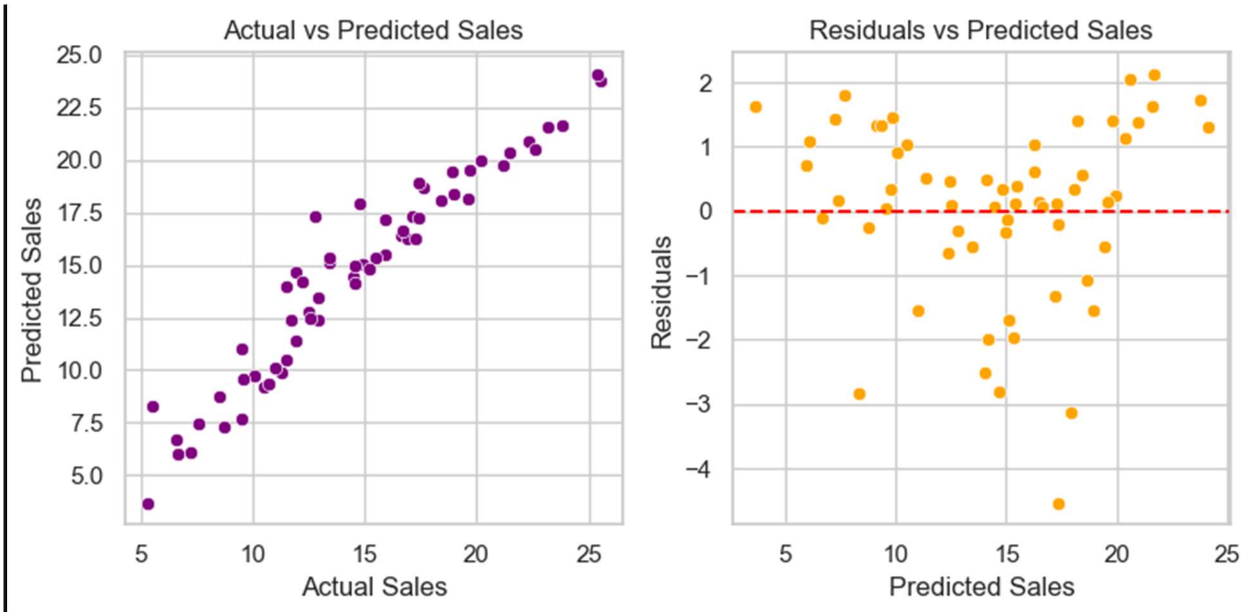
```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=1)

# Linear Regression Model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Evaluation Metrics
train_predictions = linear_model.predict(X_train)
test_predictions = linear_model.predict(X_test)

print('Training Set Metrics:')
print('R-squared:', r2_score(y_train, train_predictions))
print('MAE:', mean_absolute_error(y_train, train_predictions))
print('MSE:', mean_squared_error(y_train, train_predictions))
print('RMSE:', np.sqrt(mean_squared_error(y_train, train_predictions)))
print('\nTest Set Metrics:')
print('R-squared:', r2_score(y_test, test_predictions))
print('MAE:', mean_absolute_error(y_test, test_predictions))
print('MSE:', mean_squared_error(y_test, test_predictions))
print('RMSE:', np.sqrt(mean_squared_error(y_test, test_predictions)))

# Residuals and Predicted vs Actual
results = pd.DataFrame({'Actual Sales': y_test, 'Predicted Sales': test_predictions, 'Residuals': y_test - test_predictions})

plt.figure(figsize=(8, 4))
plt.subplot(121)
sns.scatterplot(x='Actual Sales', y='Predicted Sales', data=results, color='purple')
plt.title('Actual vs Predicted Sales')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')

plt.subplot(122)
sns.scatterplot(x='Predicted Sales', y='Residuals', data=results, color='orange')
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals vs Predicted Sales')
plt.xlabel('Predicted Sales')
plt.ylabel('Residuals')

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






Training Set Metrics:
R-squared: 0.8850071142546371
MAE: 1.3749370490107342
MSE: 3.2030691013148522
RMSE: 1.789712016307331

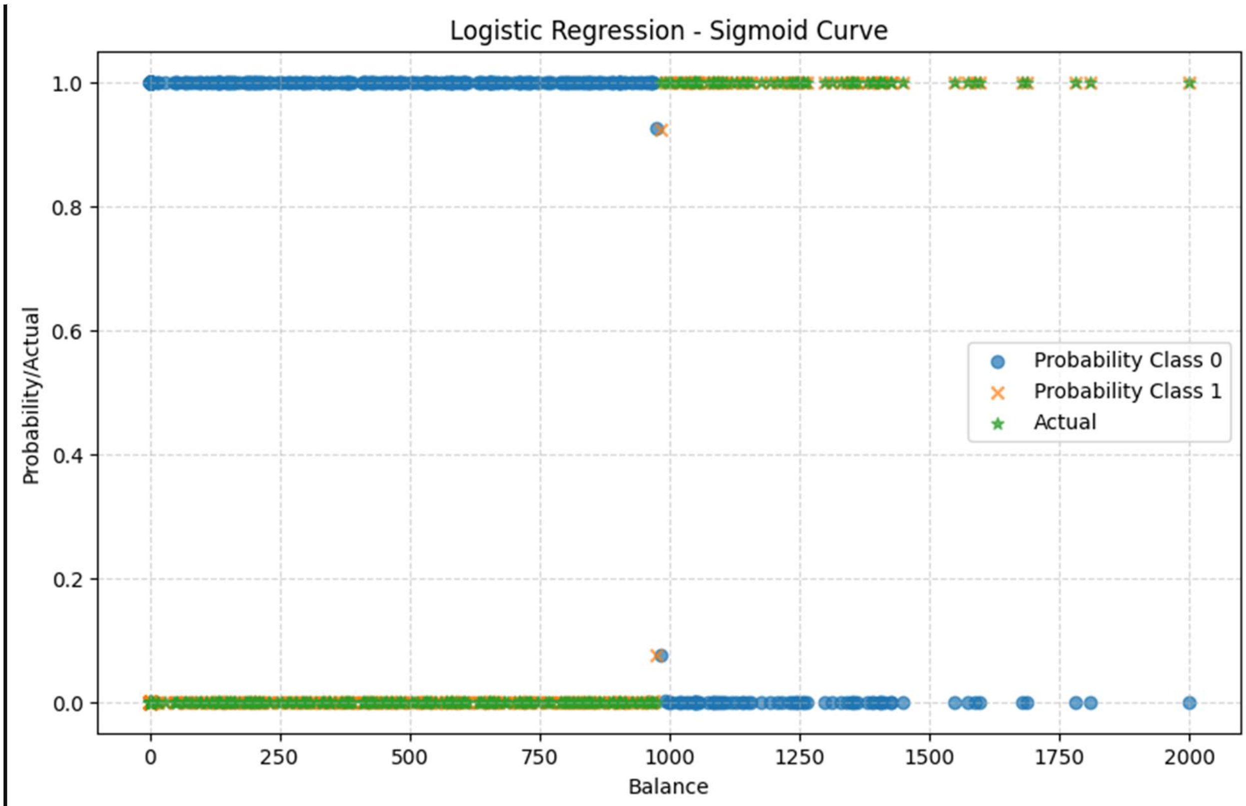
Test Set Metrics:
R-squared: 0.9225191550357023
MAE: 1.0543666890342978
MSE: 1.9274675206987768
RMSE: 1.38833264050759

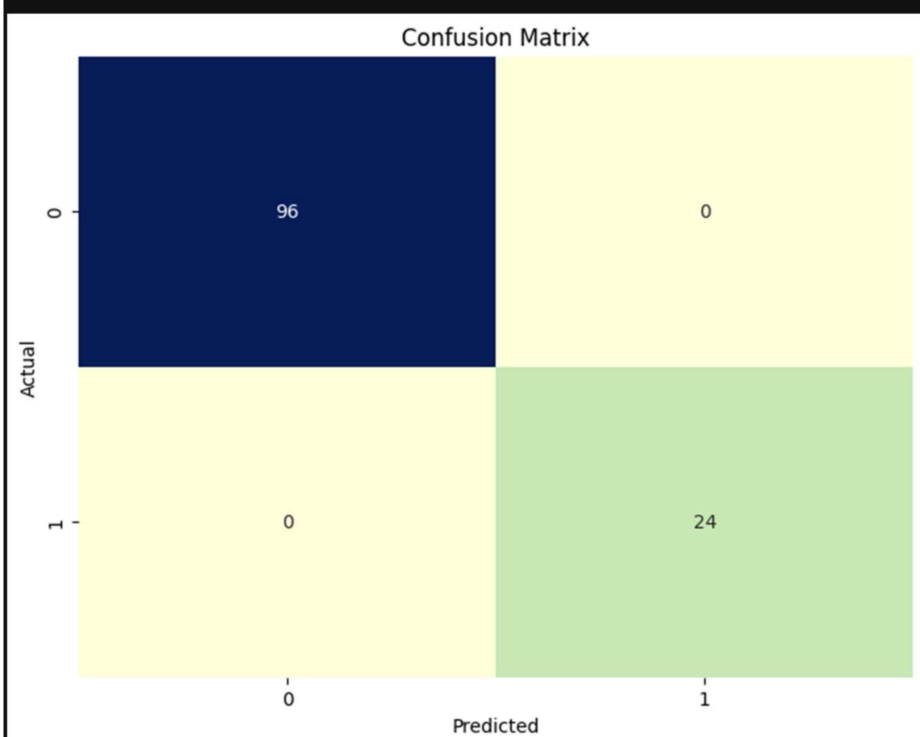



```
                    OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.885
Model:                            OLS   Adj. R-squared:                  0.882
Method:                 Least Squares   F-statistic:                     259.7
Date:                Mon, 20 Nov 2023   Prob (F-statistic):           2.39e-62
Time:                        20:09:13   Log-Likelihood:                -280.14
No. Observations:                 140   AIC:                             570.3
Df Residuals:                     135   BIC:                             585.0
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         14.0057      0.154     90.784      0.000      13.701      14.311
x1            -0.0073      0.162     -0.045      0.964      -0.329       0.314
x2             4.0210      0.153     26.334      0.000       3.719       4.323
x3             2.6148      0.165     15.885      0.000       2.289       2.940
x4             0.0389      0.173      0.225      0.822      -0.303       0.381
==============================================================================
Omnibus:                       38.658   Durbin-Watson:                   2.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.200
Skew:                          -1.242   Prob(JB):                     1.27e-16
Kurtosis:                       5.526   Cond. No.                         1.62
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.feature_selection import f_classif

# Load the credit dataset
credit_data = pd.read_csv('F:/IIT 1st Semester/Assignment/ML/3/Credit.csv')

# Preprocess data: Convert categorical to numerical
credit_data['Defaultee'] = credit_data['Defaultee'].replace(['No', 'yes'], [0, 1])
credit_data['Student'] = credit_data['Student'].map({'No': 0, 'Yes': 1})

# Apply Logistic Regression to verify sigmoid function
logistic_model = LogisticRegression()
logistic_model.fit(credit_data[['Balance']], credit_data['Defaultee'])
predicted_prob = logistic_model.predict_proba(credit_data[['Balance']])

# Visualize sigmoid curve
plt.figure(figsize=(10, 6))
plt.scatter(credit_data['Balance'], predicted_prob[:, 0], label='Probability Class 0', marker='o', alpha=0.7)
plt.scatter(credit_data['Balance'], predicted_prob[:, 1], label='Probability Class 1', marker='x', alpha=0.7)
plt.scatter(credit_data['Balance'], credit_data['Defaultee'], label='Actual', marker='*', alpha=0.7)
plt.legend()
plt.xlabel('Balance')
plt.ylabel('Probability/Actual')
plt.title('Logistic Regression - Sigmoid Curve')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()

# Logistic Regression with multiple features
features = credit_data[['Balance', 'Student', 'Income']]
target = credit_data['Defaultee']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=1)
```

```python
# Train logistic regression model
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred = logistic_model.predict(X_test)

# Model evaluation
accuracy = metrics.accuracy_score(y_test, y_pred)
classification_report = metrics.classification_report(y_test, y_pred)
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap="YlGnBu", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Perform ANOVA test for feature significance
f_values, p_values = f_classif(X_test, y_pred)

# Feature significance table
anova_results = pd.DataFrame({'Feature': X_test.columns, 'F-Value': f_values, 'P-Value': p_values})
print(anova_results)
```



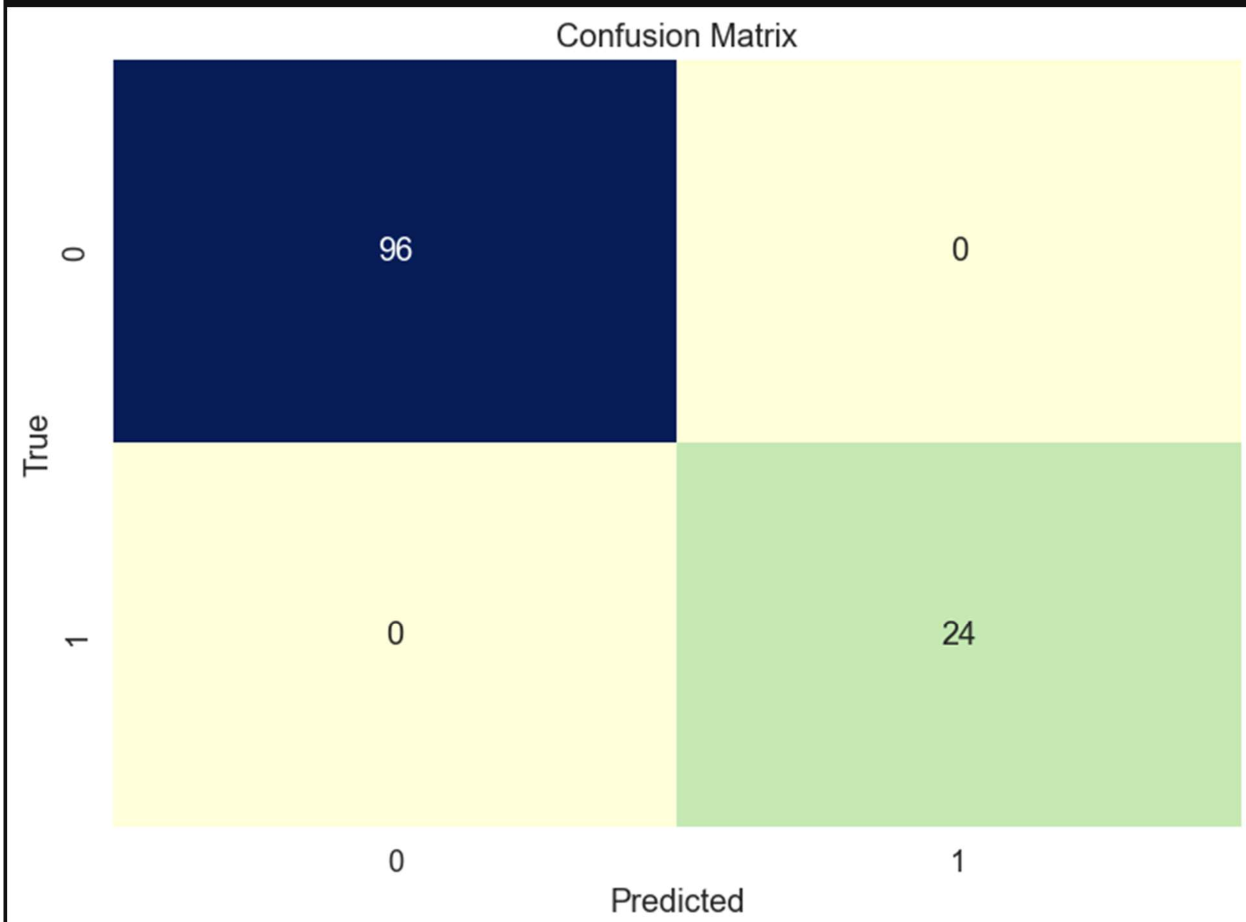




| | Feature | F-Value | P-Value |
|---|---|---|---|
| 0 | Balance | 169.903057 | 1.326267e-24 |
| 1 | Student | 6.464834 | 1.229753e-02 |
| 2 | Income | 30.587183 | 1.947321e-07 |

```python
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from mlxtend.plotting import plot_confusion_matrix
from sklearn.feature_selection import f_classif
from sklearn import metrics

# Read the data
SVM1 = pd.read_csv('F:/IIT 1st Semester/Assignment/ML/3/Credit-Modified.csv')

# Data preprocessing
SVM1['Gender'] = SVM1['Gender'].map({'Male': 0, 'Female': 1})
SVM1['Student'] = SVM1['Student'].map({'Yes': 0, 'No': 1})
SVM1['Married'] = SVM1['Married'].map({'Yes': 0, 'No': 1})

# Define features and target variable
X = SVM1.drop(['Unnamed: 0', 'Defaultee', 'Gender', 'Student', 'Married', 'Ethnicity', 'dcat'], axis=1)
y = SVM1['Defaultee']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

# Create and train the SVM model
model3 = SVC(kernel='linear', C=1.0, random_state=1)
model3.fit(X_train, y_train)

# Predict on the test set
y_pred = model3.predict(X_test)

# Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print accuracy and classification report
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(classification_rep)
```

```python
# Plotting the confusion matrix with enhanced style
fig, ax = plt.subplots(figsize=(8, 6))
sns.set(font_scale=1.2)  # Adjust font size

# Customize the heatmap colors
heatmap = sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='YlGnBu', cbar=False)
heatmap.set_xlabel('Predicted')
heatmap.set_ylabel('True')
plt.title('Confusion Matrix')

# Show plot
plt.tight_layout()
plt.show()

# Perform ANOVA test for feature significance
f_values, p_values = f_classif(X_test, y_pred)

# Create a table for feature significance
anova_results = pd.DataFrame({'Feature': X_test.columns, 'F-Value': f_values, 'P-Value': p_values})
print(anova_results)
```

```
Accuracy: 1.0000
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        96
           1       1.00      1.00      1.00        24

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```




```
       Feature     F-Value       P-Value
0       Income   30.587183  1.947321e-07
1        Limit   90.378059  2.955557e-16
2       Rating   91.075672  2.421534e-16
3        Cards    0.133908  7.150689e-01
4          Age    0.002142  9.631614e-01
5    Education    1.482109  2.258739e-01
6   Gender-num    2.754467  9.963856e-02
7  Student-num    6.464834  1.229753e-02
8      Balance  169.903057  1.326267e-24
```