A red text on a black background

Description automatically generated

**Nivrati Kothari  
25PGDM0013  
MRM Assignment**

ANOVA, Regression, Logistic Regression, CLV, Social media Extraction (Reddit and youtube), Sentiment analysis, Image face and emotion detection and mocking bot.

def add\_numbers(a, b):

return a + b

# Example usage

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

result = add\_numbers(num1, num2)

print(f"The sum of {num1} and {num2} is {result}")

**ANOVA**

import pandas as pd

import matplotlib.pyplot as plt

from io import StringIO

# Simulated data (replace with actual data if needed)

data = """Purchase Likelihood\tCondition1\tCondition2

7\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

6\tFee\thigh discount

7\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

4\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

5\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

6\tFee\thigh discount

4\tFee\thigh discount

4\tFee\thigh discount

5\tFee\thigh discount

4\tFee\thigh discount

6\tFee\thigh discount

5\tFee\thigh discount

2\tFee\tlow discount

1\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

1\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

1\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

3\tFee\tlow discount

4\tFee\tlow discount

2\tFee\tlow discount

1\tFee\tlow discount

2\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

1\tFee\tlow discount

3\tFee\tlow discount

2\tFee\tlow discount

4\tNo Fee\tlow discount

5\tNo Fee\tlow discount

3\tNo Fee\tlow discount

4\tNo Fee\tlow discount

4\tNo Fee\tlow discount

5\tNo Fee\tlow discount

3\tNo Fee\tlow discount

2\tNo Fee\tlow discount

1\tNo Fee\tlow discount

4\tNo Fee\tlow discount

3\tNo Fee\tlow discount

4\tNo Fee\tlow discount

5\tNo Fee\tlow discount

6\tNo Fee\tlow discount

4\tNo Fee\tlow discount

5\tNo Fee\tlow discount

4\tNo Fee\tlow discount

3\tNo Fee\tlow discount

2\tNo Fee\tlow discount

6\tNo Fee\tlow discount

5\tNo Fee\tlow discount

3\tNo Fee\tlow discount

4\tNo Fee\tlow discount

4\tNo Fee\tlow discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

3\tNo Fee\thigh discount

5\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

5\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

7\tNo Fee\thigh discount

5\tNo Fee\thigh discount

6\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

5\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

7\tNo Fee\thigh discount

6\tNo Fee\thigh discount

5\tNo Fee\thigh discount

5\tNo Fee\thigh discount

6\tNo Fee\thigh discount"""

# Load data into DataFrame

df = pd.read\_csv(StringIO(data), sep="\t")

# Group by conditions and calculate average purchase likelihood

average\_likelihood = df.groupby(['Condition1', 'Condition2'])['Purchase Likelihood'].mean().reset\_index()

# Display the result

print(average\_likelihood)

# Plotting

plt.figure(figsize=(10, 6))

for key, grp in average\_likelihood.groupby(['Condition1']):

plt.plot(grp['Condition2'], grp['Purchase Likelihood'], marker='o', label=key)

plt.title('Average Purchase Likelihood by Condition')

plt.xlabel('Condition2')

plt.ylabel('Average Purchase Likelihood')

plt.legend(title="Condition1")

plt.xticks(rotation=45)

plt.grid(True)

plt.tight\_layout()

plt.show()

A graph of a graph with lines

Description automatically generated with medium confidence

A white sheet with black lines

Description automatically generated

**Online Sales 2** <https://colab.research.google.com/drive/1IOr1PVDfmjAHSLuzPomMjTG8G8-ZRynh?usp=sharing>

**Online sales Logistic regression**

<https://colab.research.google.com/drive/1kAZhCArKcSxnuECD_evbRQjt2i4styHr?usp=sharing>

**REGRESSION**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import numpy as np

# Path to the dataset file

file\_path = '/content/autos regression.csv'

# Load data into DataFrame

df = pd.read\_csv(file\_path)

# Handle missing data by filling with the mode or median for numerical columns

df.fillna({'vehicleType': 'unknown', 'gearbox': 'unknown', 'notRepairedDamage': 'unknown', 'fuelType': 'unknown', 'model': 'unknown'}, inplace=True)

df['powerPS'].fillna(df['powerPS'].median(), inplace=True)

# Convert categorical variables to dummy variables

df = pd.get\_dummies(df, columns=['seller', 'offerType', 'abtest', 'vehicleType', 'gearbox', 'fuelType', 'brand', 'notRepairedDamage', 'model'], drop\_first=True)

# Define dependent and independent variables

X = df.drop(['price', 'index', 'name', 'dateCrawled', 'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen'], axis=1)

y = df['price']

# Split the data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the regression model

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# Predict the prices

y\_pred = regressor.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Display coefficients

coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': regressor.coef\_})

print(coefficients.sort\_values(by='Coefficient', ascending=False))

# Assumptions:

# 1. Linearity - The relationship between the DV and IVs is linear.

# 2. No multicollinearity - Categorical encoding may introduce collinearity.

# 3. Homoscedasticity - Residuals have constant variance.

# 4. Normality of residuals - Assumed normal distribution of errors.

import pandas as pd

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.metrics import mean\_squared\_error, r2\_score

import numpy as np

# Path to the dataset file

file\_path = '/content/autos regression.csv'

# Load data into DataFrame

df = pd.read\_csv(file\_path)

# Handle missing data

# Fill numerical columns with median and categorical columns with mode

df['powerPS'].fillna(df['powerPS'].median(), inplace=True)

df['yearOfRegistration'].fillna(df['yearOfRegistration'].median(), inplace=True)

df.fillna({'vehicleType': 'unknown', 'gearbox': 'unknown', 'notRepairedDamage': 'unknown', 'fuelType': 'unknown', 'model': 'unknown'}, inplace=True)

# Convert categorical variables to dummy variables

df = pd.get\_dummies(df, columns=['seller', 'offerType', 'abtest', 'vehicleType', 'gearbox', 'fuelType', 'brand', 'notRepairedDamage', 'model'], drop\_first=True)

# Define dependent and independent variables

X = df.drop(['price', 'index', 'name', 'dateCrawled', 'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen'], axis=1)

y = df['price']

# Split the data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the regression model

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# Predict the prices

y\_pred = regressor.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

print(f'R-squared: {r2}')

# Display coefficients

coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': regressor.coef\_})

print(coefficients.sort\_values(by='Coefficient', ascending=False))

# Visualization of regression results

plt.figure(figsize=(10, 6))

sns.scatterplot(x=y\_test, y=y\_pred)

plt.xlabel('Actual Prices')

plt.ylabel('Predicted Prices')

plt.title('Actual vs Predicted Prices')

plt.grid(True)

plt.show()

# Visualization of residuals

residuals = y\_test - y\_pred

plt.figure(figsize=(10, 6))

sns.histplot(residuals, kde=True)

plt.xlabel('Residuals')

plt.title('Distribution of Residuals')

plt.grid(True)

plt.show()

# Assumptions:

# 1. Linearity - The relationship between the dependent and independent variables is assumed linear.

# 2. No multicollinearity - Categorical encoding can introduce collinearity; VIF checks may be required.

# 3. Homoscedasticity - Constant variance of residuals across predictions.

# 4. Normality of residuals - Residuals are assumed to be normally distributed.

**CLV**  
<https://colab.research.google.com/drive/1nzttbmTX6BjlP4BCrdm7UTLpd6Me3qbe?usp=sharing>

**LOGISTIC REGRESSION**

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 LogisticRegression

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix, roc\_curve, auc

# Step 1: Load data from file

file\_path = '/content/bank marketing - logistic.xlsx' # Path to the Excel file

def load\_data(file\_path):

return pd.read\_excel(file\_path)

# Step 2: Preprocess data

def preprocess\_data(data):

# Identify categorical variables

categorical\_columns = data.select\_dtypes(include=['object']).columns.tolist()

categorical\_columns.remove('y') # Exclude the dependent variable

# Convert categorical variables to dummy variables

data = pd.get\_dummies(data, columns=categorical\_columns, drop\_first=True)

# Convert dependent variable 'y' to binary (1 for 'yes', 0 for 'no')

data['y'] = data['y'].apply(lambda x: 1 if x == 'yes' else 0)

# Split data into independent (X) and dependent (y) variables

X = data.drop(['y'], axis=1) # All variables except 'y'

y = data['y'] # Target variable

return X, y

# Step 3: Train logistic regression model

def train\_logistic\_regression(X, y):

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Initialize logistic regression model

model = LogisticRegression(max\_iter=1000) # Increase max\_iter to ensure convergence

# Train the model

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

y\_prob = model.predict\_proba(X\_test)[:, 1] # Probabilities for ROC curve

# Evaluate the model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

return model, y\_test, y\_pred, y\_prob

# Step 4: Visualization and Performance Evaluation

def visualize\_results(y\_test, y\_pred, y\_prob):

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# ROC Curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_prob)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC Curve (area = {roc\_auc:.2f}')

plt.plot([0, 1], [0, 1], linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()

# Step 5: Main function to execute the steps

def main():

data = load\_data(file\_path)

# Preprocess the data

X, y = preprocess\_data(data)

# Train the logistic regression model

model, y\_test, y\_pred, y\_prob = train\_logistic\_regression(X, y)

# Display the coefficients of the logistic regression model

print("Feature Coefficients:\n")

for feature, coef in zip(X.columns, model.coef\_[0]):

print(f"{feature}: {coef}")

# Visualize results

visualize\_results(y\_test, y\_pred, y\_prob)

# Execute the main function

if \_\_name\_\_ == "\_\_main\_\_":

main()

A graph of a curve

Description automatically generated

A blue squares with white text

Description automatically generated

A graph of a number of transactions

Description automatically generated

**Youtube Sentimental Analysis**

<https://colab.research.google.com/drive/1-BWO1ws6-Sq1xBmLjAmMX9sRs41xWvSm?usp=sharing>

**Image Understanding**

<https://colab.research.google.com/drive/1gSsI901biJQ1Rjbe-5YLkXxvSbT7sl62?usp=sharing>

**Reddit Analysis**

<https://colab.research.google.com/drive/139Ww_YYtXM2I86gwye4Z6xuPDGSaITto?usp=sharing>