# Part 1: Mutual Information Classification

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, mutual_info_classif

loan_data=pd.read_csv("loan.csv")

loan_data.head()
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

```
loan_data.isnull().sum()
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
# Step 2: Apply necessary processing
# Label encoding for the target variable and handling null values
label_encoder = LabelEncoder()
loan_data['Loan_Status'] =
label_encoder.fit_transform(loan_data['Loan_Status'])


# One-hot encoding for all categorical features
loan_data = pd.get_dummies(loan_data, drop_first=True)

# Handling null values (you might need to customize this based on your
dataset)
loan_data = loan_data.fillna(0)  # Filling null values with 0 for
simplicity

# Step 3: Separate features (X) and target variable (y: Loan_Status)
X = loan_data.drop('Loan_Status', axis=1)  # Features
y = loan_data['Loan_Status']  # Target variable

# Step 4: Use SelectKBest for feature selection
k_best_features = 5  # Choose an appropriate value of K (number of
features to select)

# Initialize SelectKBest with mutual information classification
feature_selector = SelectKBest(score_func=mutual_info_classif,
k=k_best_features)

# Fit the feature selection model and transform the feature matrix
accordingly
X_selected = feature_selector.fit_transform(X, y)

# Get the indices of the selected features
selected_feature_indices = feature_selector.get_support(indices=True)

# Print the names or indices of the selected features
selected_feature_names = X.columns[selected_feature_indices]
print(f"Selected Features: {selected_feature_names}")
```

```
Selected Features: Index(['Credit_History', 'Loan_ID_LP001384',
'Loan_ID_LP002284',
       'Loan_ID_LP002317', 'Loan_ID_LP002862'],
      dtype='object')
```

# Part 2: Mutual Information Regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest,
mutual_info_regression
from sklearn.impute import SimpleImputer
```

```
from sklearn.preprocessing import LabelEncoder


# Load the dataset
df = pd.read_csv('Housing.csv')

# Separate features (X) and target variable (y)
X = df.drop('SalePrice', axis=1)  # Assuming 'Price' is the target
variable
y = df['SalePrice']

# Handle categorical features with label encoding
label_encoder = LabelEncoder()
X_categorical = X.select_dtypes(include='object')
X_categorical_encoded =
X_categorical.apply(label_encoder.fit_transform)
X[X_categorical.columns] = X_categorical_encoded

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y,
test_size=0.2, random_state=42)

# Use SelectKBest to select the top K features based on mutual
information scores
k = 5  # Choose an appropriate value of K
selector = SelectKBest(score_func=mutual_info_regression, k=k)


# Fit the feature selection model on the dataset and transform the
feature matrix
X_selected = selector.fit_transform(X_train, y_train)

# Get the names or indices of the selected features
selected_feature_indices = selector.get_support(indices=True)
selected_feature_names = X.columns[selected_feature_indices]


# Print the names or indices of the selected features
print("Selected Feature Names:", selected_feature_names)
print("Selected Feature Indices:", selected_feature_indices)
```

```
Selected Feature Names: Index(['Neighborhood', 'OverallQual',
'TotalBsmtSF', 'GrLivArea',
       'GarageArea'],
      dtype='object')
Selected Feature Indices: [12 17 38 46 62]
```

# Part 3 : Linear Regression on the Housing Dataset

```
# Step 1: Load the Housing dataset using the pandas library
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

housing_data=pd.read_csv("Housing.csv")

# Step 2: Apply necessary preprocessing steps
# Assuming 'X' has categorical features, perform one-hot encoding
X_encoded = pd.get_dummies(X, drop_first=True)  # drop_first=True to
avoid the dummy variable trap

# Impute missing values using the mean strategy
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X_encoded)

# Separate the features (X) and the target variable (y) from the
encoded data
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y,
test_size=0.2, random_state=42)


# Fit a linear regression model to the training data
linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train, y_train)

LinearRegression()

# Predict house prices for the testing data
y_pred = linear_reg_model.predict(X_test)

# Evaluate the performance of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the MSE and R^2 values to assess the model's accuracy
print("Mean Squared Error (MSE):", mse)
print("Coefficient of Determination (R^2):", r2)

Mean Squared Error (MSE): 2419721731.0240765
Coefficient of Determination (R^2): 0.6845347035231972
```

```
# Plot a scatter plot between the predicted and actual house prices
plt.scatter(y_test, y_pred)
plt.xlabel("Actual House Prices")
plt.ylabel("Predicted House Prices")
plt.title("Scatter Plot of Actual vs Predicted House Prices")
plt.show()
```