



MACHINE LEARNING CLASSIFICATION PROJECT



CREDIT RISK PREDICTION

TEAM-2

G. MADHU KIRAN

CHIRAMJEEVI

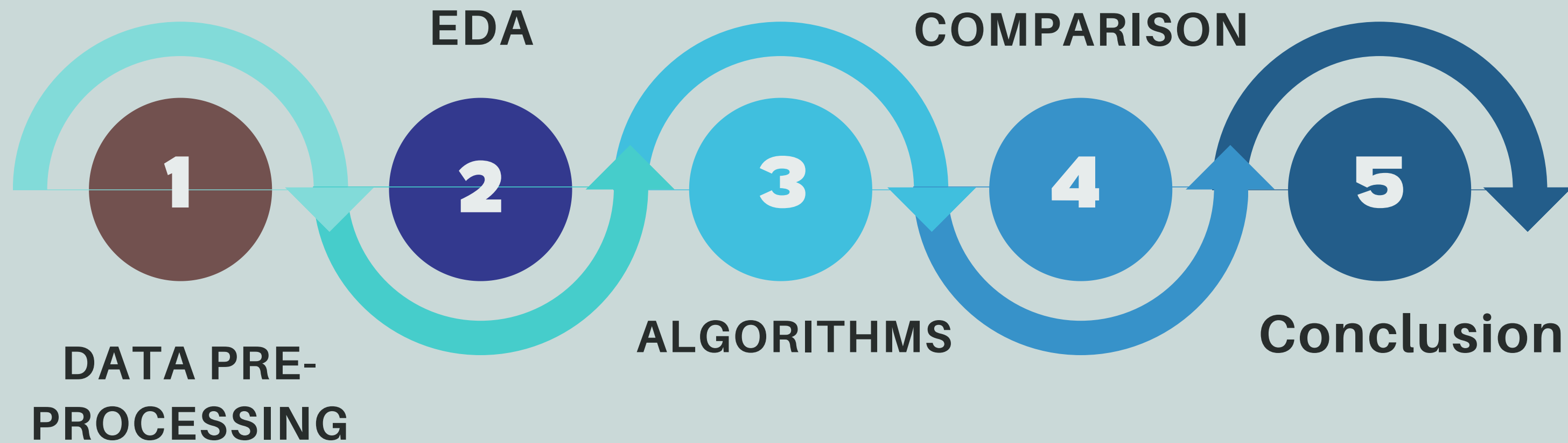
P SHRAVANI

SATHVIKA REDDY



BHAVAN'S VIVEKANANDA COLLEGE | BSC HONS DATA SCIENCE

MACHINE LEARNING PROJECT STAGES



ABOUT THE DATA:

The task here is to build a classification model to predict "**TARGET_credit_risk**".

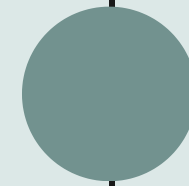
OBJECTIVE:

The original dataset contains 1000 entries with 20 categorical/symbolic attributes. Each person is classified as good or bad credit risks according to the set of attributes. We have to predict the credit risk of a person.

THE PATH:

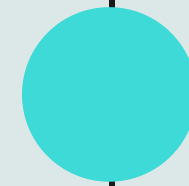
Implementing different classification models to fit the best one to the dataset. We followed a traditional approach with dummy variable encoding

DATA AND DATA QUALITY CHECK:



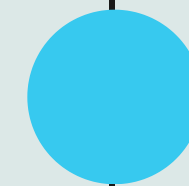
INTRODUCTION

Brief about the instances and attributes



VARIABLES

Detailed description about variables



DATA PRE PROCESSING

Data Cleaning, Data Transformation etc

INTRODUCTION:

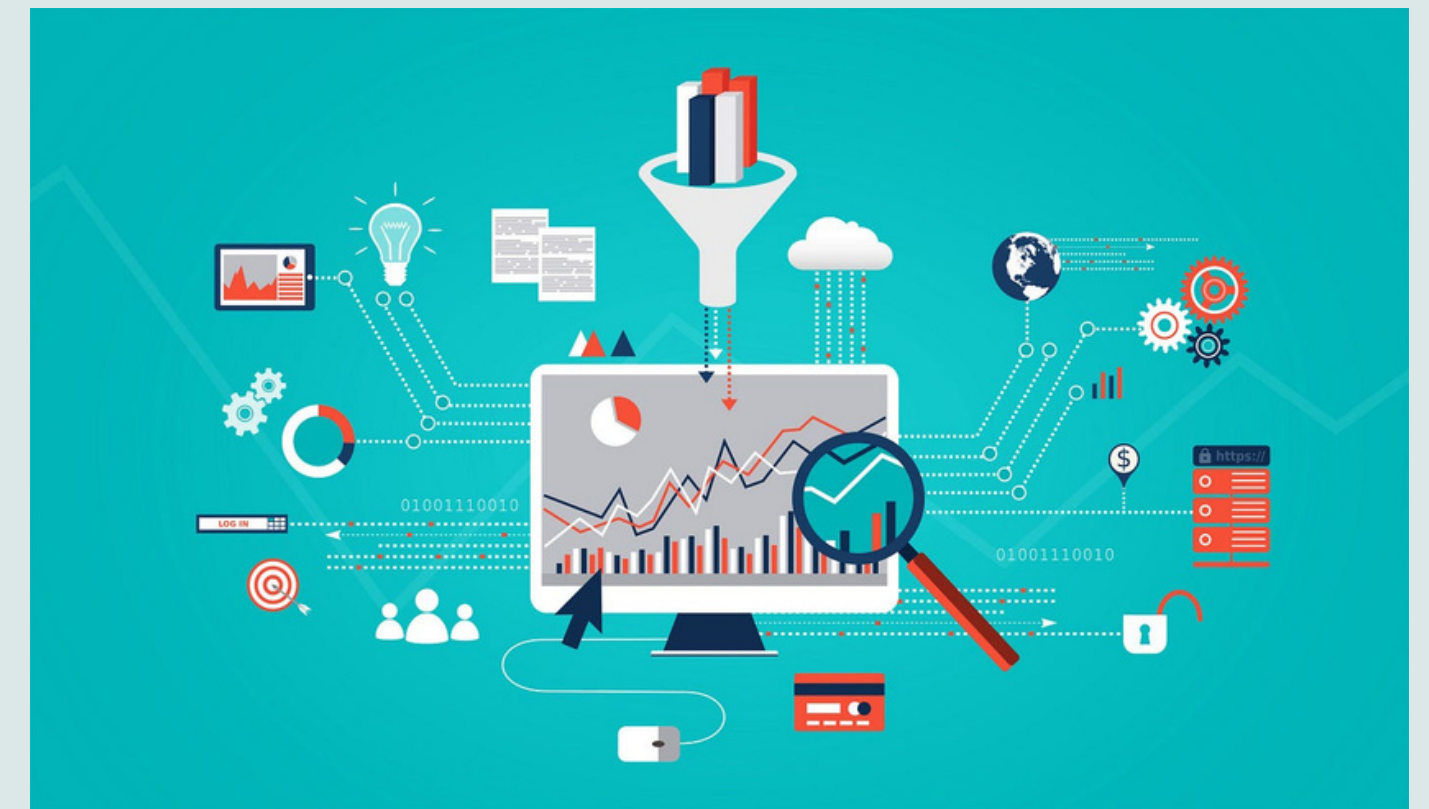
Number of instances	1000
Number of Attributes	21
Attribute breakdown	20 quantitative input variables, and 1 quantitative output variable
Missing Attribute Values	None

VARIABLES:

TARGET_creditrisk	Has the credit contract been complied with (good) or not (bad)
status	status of the debtor's checking account with the bank
duration	duration in months
credit_history	history of compliance with previous or concurrent credit contracts
purpose	purpose for which the credit is needed
saving	debtor's savings
employment_duration	duration of debtor's employment with current employer

DATA PRE-PROCESSING:

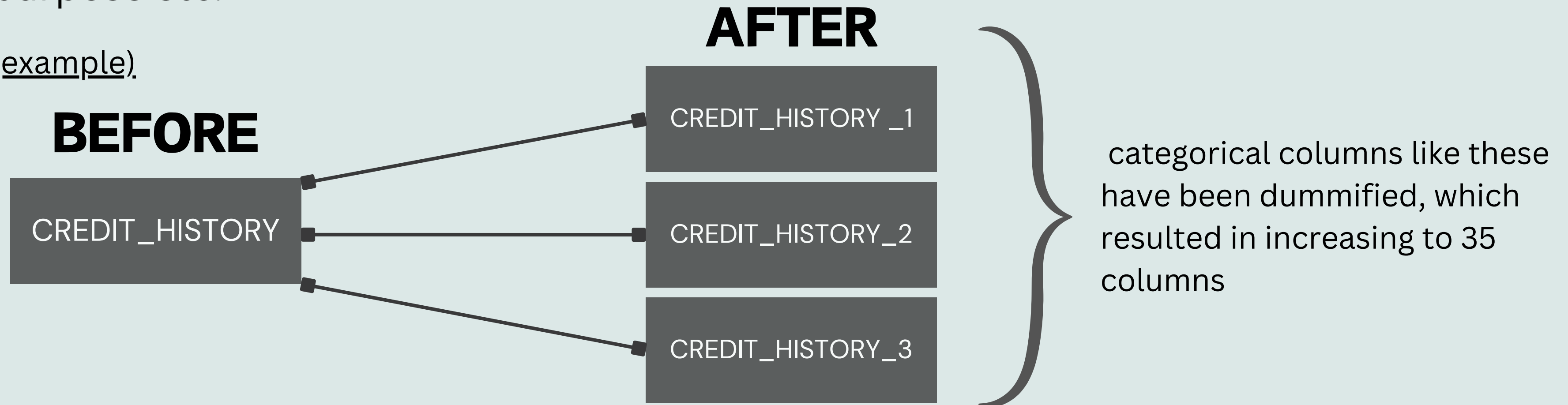
1. There are few columns in the dataset which are categorical but they're coded as 0's and 1's.
2. There are no null values in the data set.
3. There are No duplicate values in the dataset.



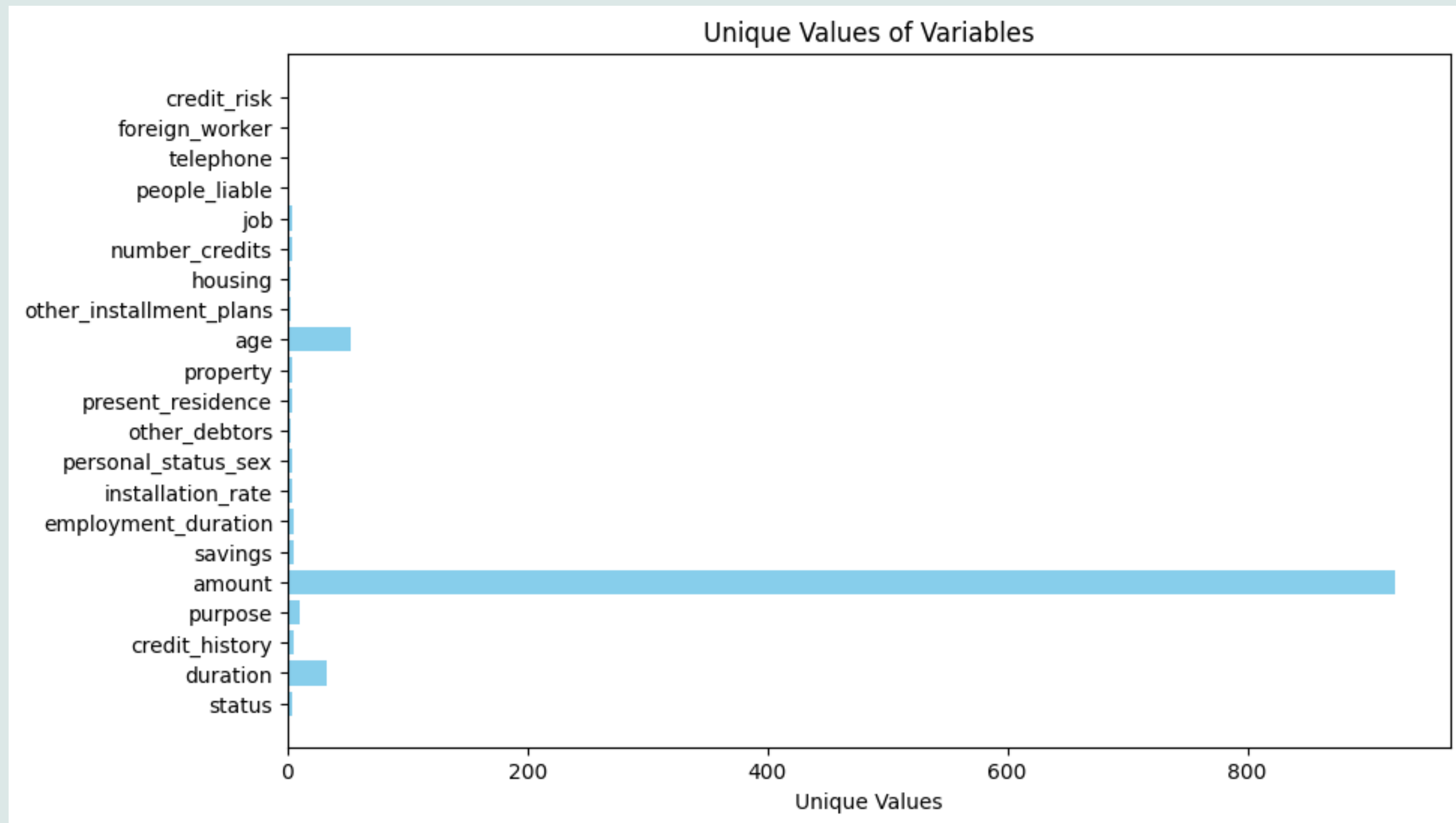
DUMMY VARIABLE ENCODING:

1. Before dummy variable encoding, there were 21 columns
2. After dummy variable encoding, resultant obtained to 35 columns
3. Categorical columns are status, credit history, personal status sex, other debtors, purpose etc.

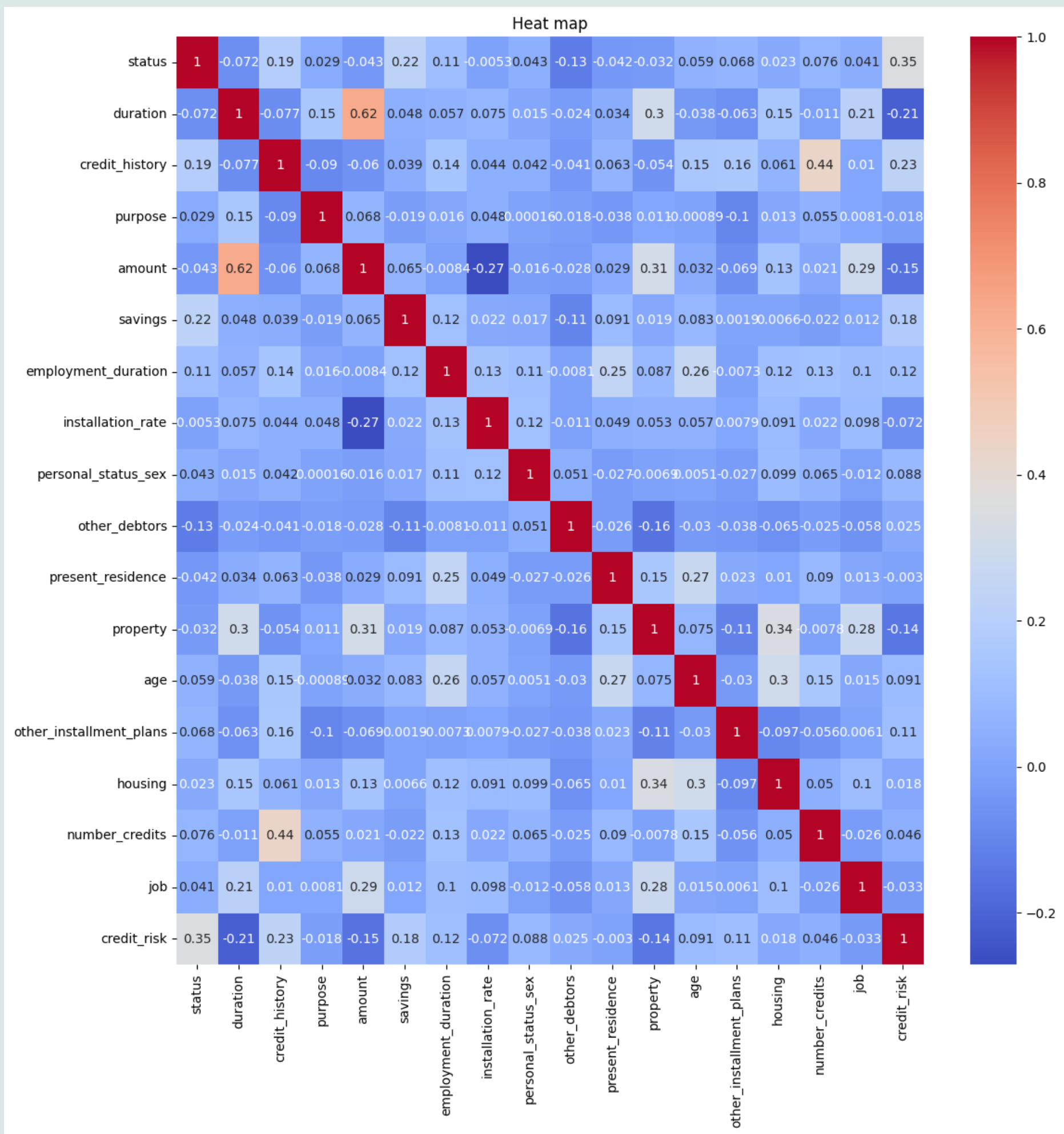
(example).



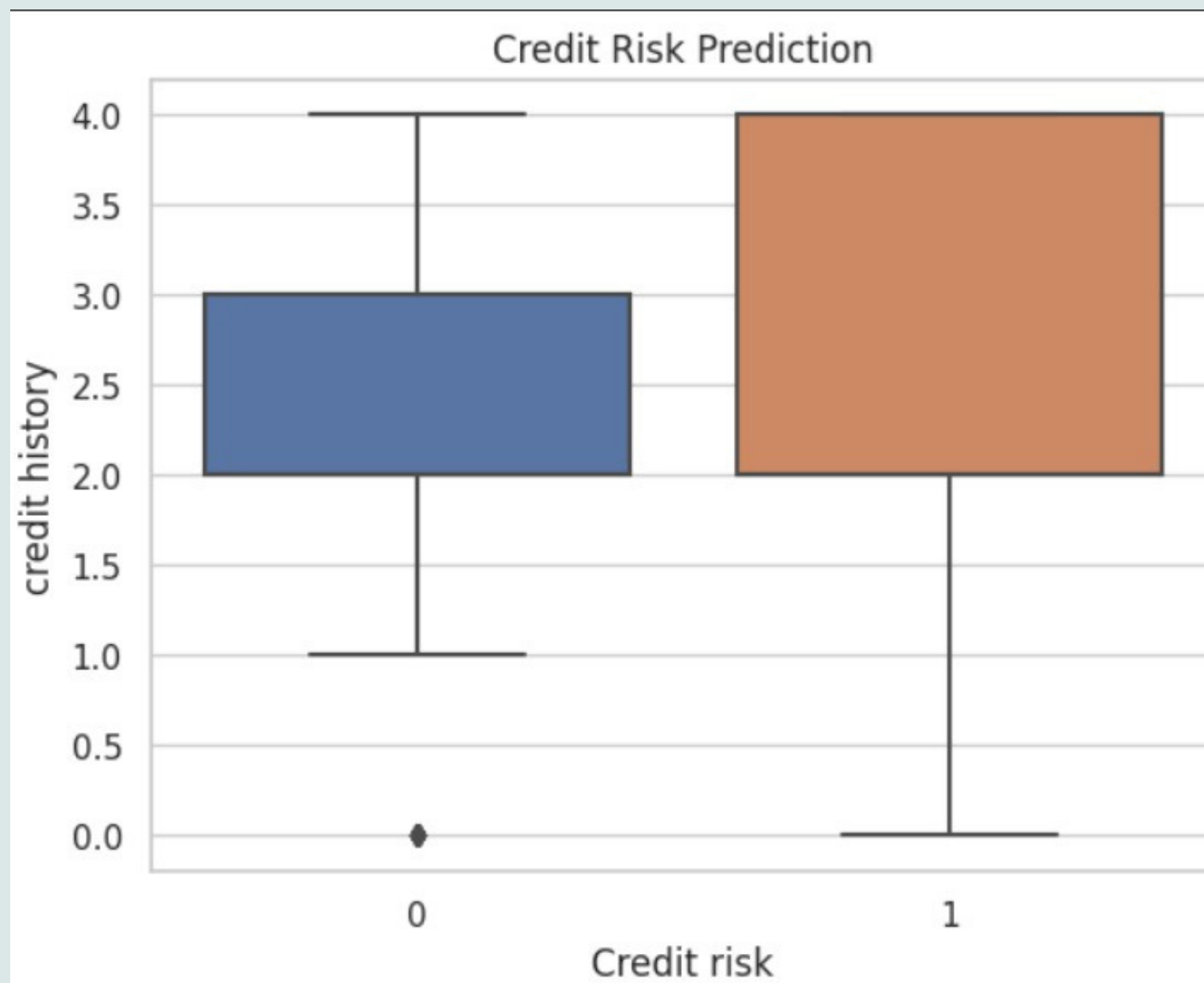
EXPLORATORY DATA ANALYSIS:



Graph represents the unique values of each variable

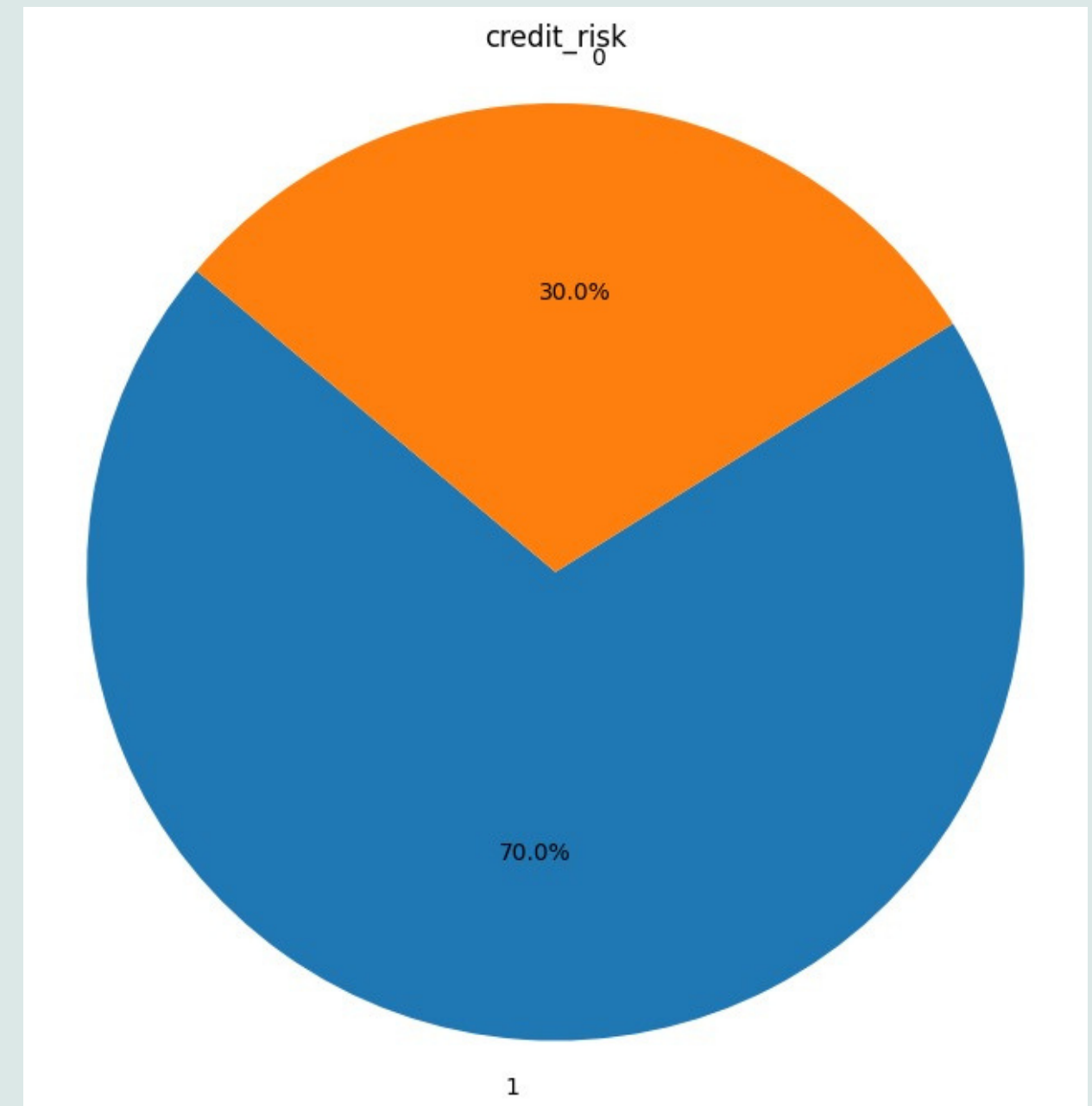


- Heatmap represents the correlation between different variables.
- There is a positive correlation between duration and amount



- **Box Plot** to represent outliers.

- **Pie Chart** representing the 700 good and 300 bad credits with 20 predictor variables.



VARIANCE INFLATION FACTOR

MULTICOLLINEARITY ANALYSIS

Variables with Low VIF:

- purpose_1 to purpose_10, telephone, other_debtors_2, other_debtors_3, other_installment_plans_2, other_installment_plans_3, housing_3, etc. (VIF values less than 5)

VARIABLES	VIF
duration	8.121870
amount	5.747600
savings	3.163318
employment_duration	11.026173
installation_rate	11.056842
present_residence	9.727082
property	9.850866
age	13.796849
number_credits	10.236151

MODEL BUILDING:

1. DECISION TREE CLASSIFIER:

TRAIN_TEST RATIO	ACCURACY
65-35	69%
70-30	68%
75-25	71.2%
80-20	71.5%

2. K NEIGHBORS CLASSIFIER:

TRAIN_TEST RATIO	ACCURACY
65-35	88%
70-30	86%
75-25	87%
80-20	85%

3. SUPPORT VECTOR CLASSIFIER:

TRAIN_TEST RATIO	ACCURACY
65-35	94%
70-30	94.3%
75-25	94.8%
80-20	95%

4. GRADIENT BOOSTING:

TRAIN_TEST RATIO	ACCURACY
65-35	89%
70-30	87%
75-25	90%
80-20	91%

5. NAIVE BAYES CLASSIFIER:

TRAIN_TEST RATIO	ACCURACY
65-35	94%
70-30	95%
75-25	96.4%
80-20	96%

6. RANDOM FOREST CLASSIFIER:

TRAIN_TEST RATIO	N_ESTIMATORS	ACCURACY
65-35	100	97.4%
70-30	100	98.6%
75-25	100	98.8%
80-20	100	97.5%

7. NEURAL NETWORK:

TRAIN_TEST RATIO	ARCHITECTURE	OPTIMIZER	ACCURACY	EPOCHS
65-35	128-64-1	Adam	98.2%	100
70-30	128-64-1	Adam	97%	100
75-25	128-64-1	Adam	96%	100
80-20	128-64-1	Adam	96.2%	100

COMPARISON OF
MODELS:

MODEL	ACCURACY(Test Size 75-25)
Decision Tree Classifier	71.2%
K Neighbors Classifier	87%
Support Vector Classifier	94.8%
Gradient Boosting	90%
Naive Bayes	96.4%
Random Forest Classifier	98.8%
Neural Network	96%

CONCLUSION:

1. For this Credit dataset, we applied 7 algorithms. Decision Tree, K Neighbors, Support Vector, Gradient Boosting, Random Forest, Naive Bayes, Neural Network.
2. Among all the important variables, Credit_History is the most impactful to the target variable Credit_Risk. (Based on the past credit history, we can predict potential risks which may occur in near future by deploying different classification models)
3. Here, we conclude that Random Forest Classifier is the best model for Credit Dataset with 98.8% accuracy, test size of 75-25 ratio



THANK YOU

TEAM-2
G. MADHU KIRAN
CHIRAMJEEVI
P. SHRAVANI
SATHVIKA REDDY

APPENDIX

TRAINING AND TESTING:

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

CHECKING ACCURACY:

```
print("accuracy", metrics.accuracy_score(y_test, y_pred))
```


DUMMY VARIABLE ENCODING:

```
] categorical_columns = ["status", "credit_history", "purpose", "personal_status_sex", "other_debtors", "other_insta
# Renameing the column 'saving' to avoid keyword conflict
df.rename(columns={'saving': 'savings'}, inplace=True)
# Perform dummy variable encoding for each categorical column
for column in categorical_columns:
    dummy_df = pd.get_dummies(df[column], prefix=column, drop_first=True)
    df = pd.concat([df, dummy_df], axis=1)
# Drop the original categorical columns
df = df.drop(categorical_columns, axis=1)
print("\nDataFrame after encoding:")
print(df)
```

DECISION TREE CLASSIFIER:

```
model=DecisionTreeClassifier()  
model.fit(X_train,y_train)  
y_pred=model.predict(X_test)  
y_pred
```

K NEIGHBORS CLASSIFIER:

```
clf=KNeighborsClassifier(n_neighbors=5)  
clf.fit(X_train,y_train)
```

▼ KNeighborsClassifier

KNeighborsClassifier()

SUPPORT VECTOR CLASSIFIER:

```
cls=SVC(kernel='linear')
```

```
cls.fit(X_train,y_train)
```

▼ SVC

```
SVC(kernel='linear')
```

GRADIENT BOOSTING:

```
clf=GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
clf.fit(X_train,y_train)
```

▼ GradientBoostingClassifier

GradientBoostingClassifier(random_state=42)

NAIVE BAYES:

```
# Initialize the Gaussian Naive Bayes classifier
naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(X_train,y_train)
```

▼ GaussianNB

GaussianNB()

RANDOM FOREST CLASSIFIER:

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)  
rf_classifier.fit(X_train,y_train)
```

▼ RandomForestClassifier

```
RandomForestClassifier(random_state=42)
```

NEURAL NETWORK:

```
# Define the ANN model
model = Sequential()
model.add(Dense(128, activation='relu', input_dim=X_train.shape[1]))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32)

# Evaluate the model on the test set
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```


VARIANCE INFLATION FACTOR:

```
# Import library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(df):
    vif = pd.DataFrame()
    vif["variables"] = df.columns
    vif["VIF"] = [variance_inflation_factor(df.values,i) for i in range((df).shape[1])]
    return(vif)
calc_vif(df)
```

28	purpose_9	1.609290
29	purpose_10	1.169066
30	personal_status_sex_2	6.417080
31	personal_status_sex_3	11.066844
32	personal_status_sex_4	2.647124
33	other_debtors_2	1.121344
34	other_debtors_3	1.180528
35	other_installment_plans_2	1.390659
36	other_installment_plans_3	7.620601
37	housing_2	5.969744
38	housing_3	2.305738