Homework 4 Report

Ezequiel Buck, Esteban Murillo

September 29, 2025

1 Part A: Data Preprocessing and Feature Creation

1.1 Key Modifications

1. **Exclude Status X**: All entries with Status = X are ignored because customers with no loans are neither "good" nor "bad" at paying their loans.

2. **Compute Status Counts**: For each customer, we calculate the total counts of "C", "0", and "1" statuses, as well as total observations (excluding Xs).

3. **Calculate Fractions**: We compute the fraction of observations that are "C", "0", and "1" for each customer:

```
# Group by ID and calculate fractions for each status
status_fractions = credit_df[credit_df['STATUS'].isin(['C', '0', '1'])].groupby(
    credit_id_col)['STATUS'].value_counts(normalize=True).unstack(fill_value=0)

# Rename columns to indicate they are fractions
status_fractions.columns = [f'{col}_fraction' for col in status_fractions.columns]

# Add these new columns to app_df
app_df = app_df.join(status_fractions, on=app_id_col)
```

4. **Delinquency Label**: Compute the "Delinquent" column as before "1" if customer has any statuses equal to "2", "3", "4", or "5", and zero otherwise:

```
def is_delinquent(customer_id):
```

```python
    # Define what counts as a delinquent status
    delinquent_statuses = ['2', '3', '4', '5', 2, 3, 4, 5]

    # Get all status records for this customer
    customer_records = credit_df[credit_df[credit_id_col] == customer_id]
    customer_statuses = customer_records['STATUS']

    # Check if customer has any delinquent status
    has_delinquent_status = any(
        status in delinquent_statuses for status in customer_statuses)

    # Return 1 if delinquent, 0 if not
    return int(has_delinquent_status)

# Apply the function to each record
app_df['Delinquent'] = app_df[app_id_col].apply(is_delinquent)
```

1.2 Terminal Output

The program produces the following output showing the data processing:

```
Value counts of the STATUS column: STATUS
C    442031
0    383120
1     11090
5      1693
2       868
3       320
4       223

Total observations in the credit_df: ID
5005005    61
5022730    61
5061848    61
5061810    61
```

```
5061741    61
           ..
5060177     1
5099734     1
5060155     1
5035790     1
5135846     1

Delinquent counts:
 Delinquent
0    32494
1      616
```

2 Part B: Feature Exploration

2.1 Histograms of Ratio Columns

We created histograms to show the distribution of the new ratio columns:

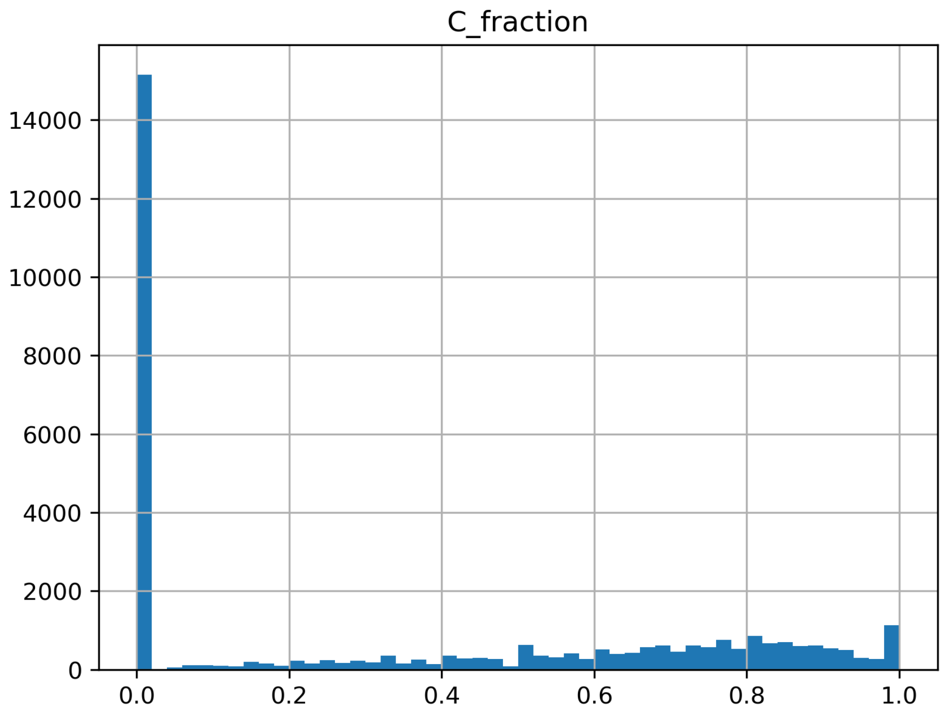

Figure 1: Histogram of C_fraction (Closed Account Fractions)

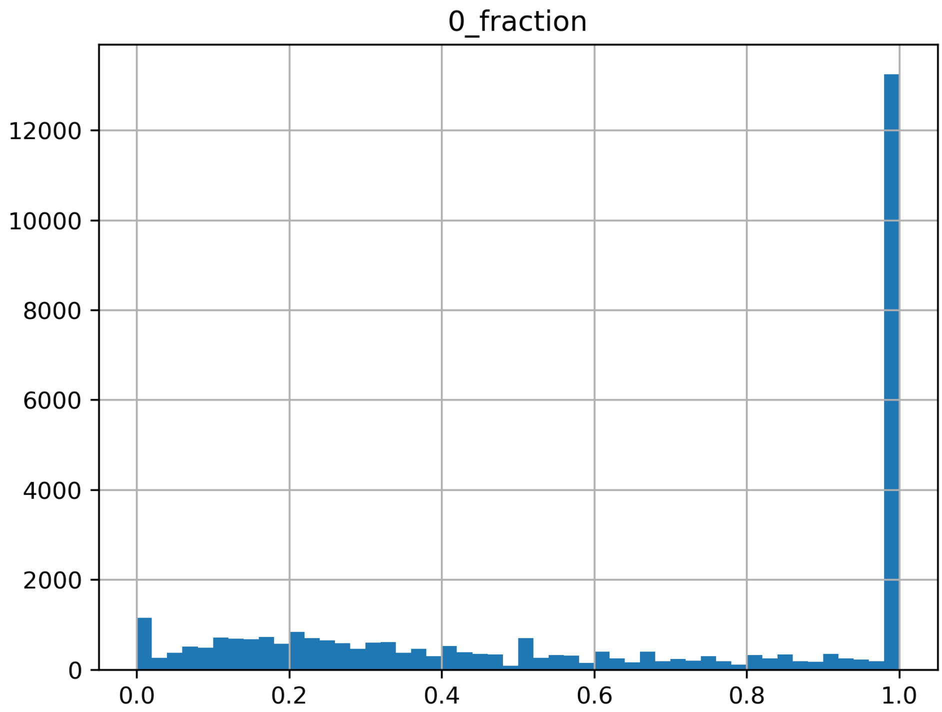

Figure 2: Histogram of 0_fraction (No Payment Due Fractions)

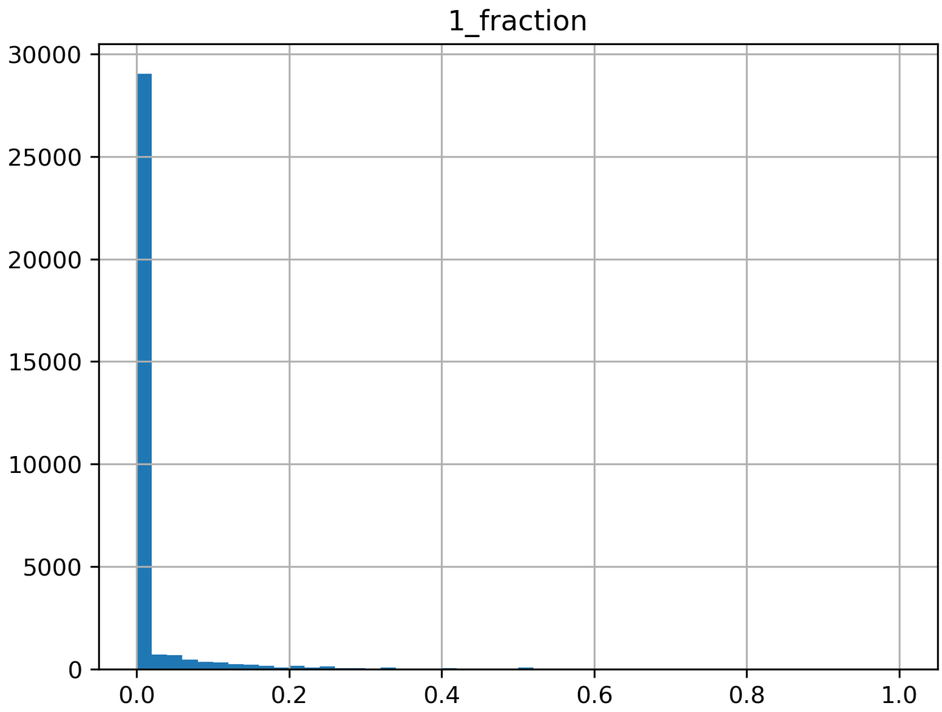

Figure 3: Histogram of 1_fraction (Current Payment Fractions)

2.2 Bar Plots of Delinquency Rates

We created bar plots showing delinquency rates for customers in different intervals of these ratio columns:

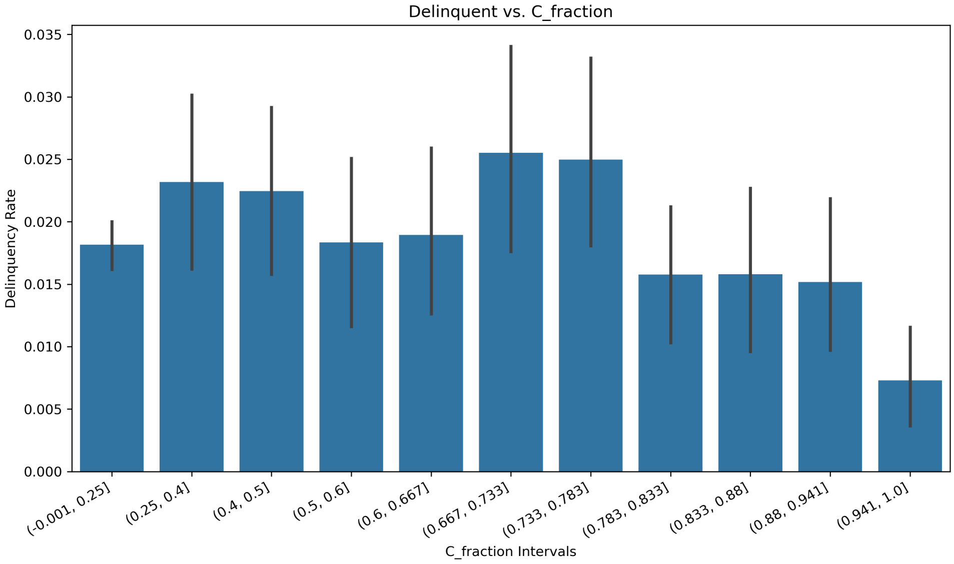

Figure 4: Delinquency Rate vs. C_fraction Intervals

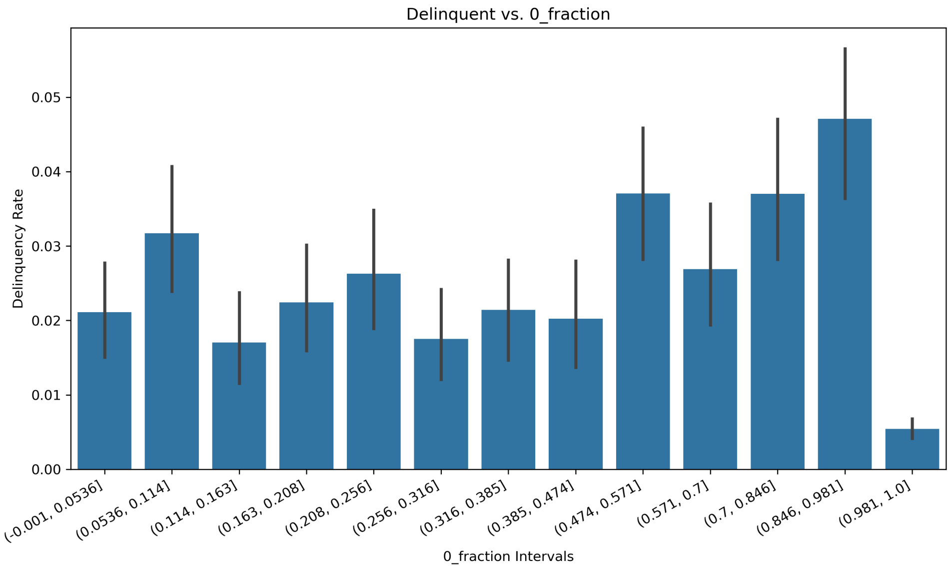

Figure 5: Delinquency Rate vs. 0_fraction Intervals

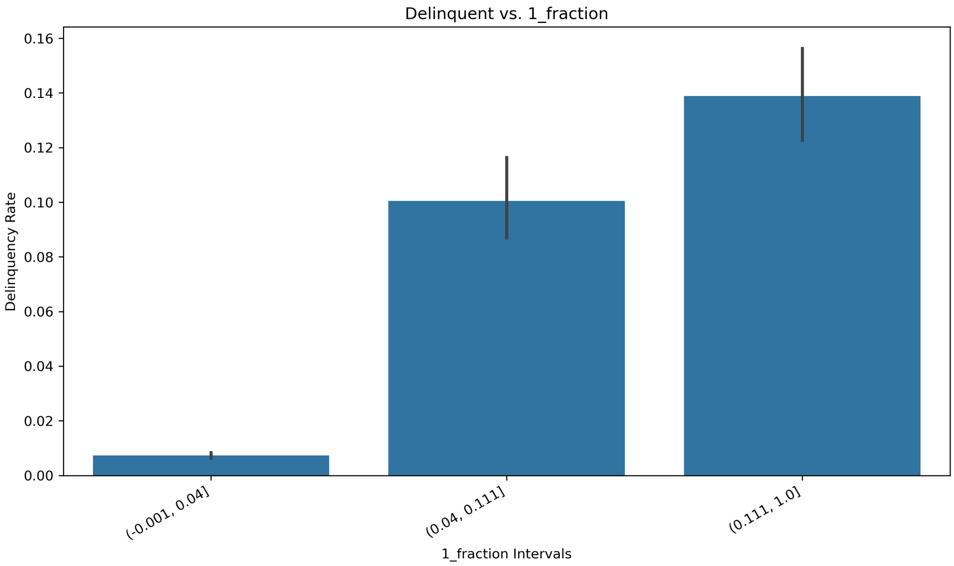

Figure 6: Delinquency Rate vs. 1_fraction Intervals

2.3 Code for Feature Exploration

```
# Plot bar chart of the C, 0, 1 ratios of the STATUS column
for col in fractions:
    df.hist(column=[col], bins=50)
    plt.title(col)
    plt.savefig(f'graphs/{col}_histogram.png', dpi=300, bbox_inches='tight')
    plt.show()

# Plot a bar chart of the Delinquent vs. the fractions
for col in fractions:
    plt.figure(figsize=(10, 6))
    df[col] = pd.qcut(df[col], 20, duplicates="drop")
    lm = sns.barplot(data=df, x=col, y="Delinquent")
    plt.xticks(rotation=30, ha='right')
    plt.title(f"Delinquent vs. {col}")
    plt.xlabel(f"{col} Intervals")
    plt.ylabel("Delinquency Rate")
    plt.tight_layout()
    plt.savefig(f'graphs/delinquent_vs_{col}.png', dpi=300, bbox_inches='tight')
    plt.show()
```

3 Part C: Model Training and Evaluation

3.1 Logistic Regression Model

```python
df = pd.get_dummies(df, prefix_sep="_", drop_first=False, dtype=int)
labels = df["Delinquent"]
df = df.drop(columns="Delinquent")

# Shuffle and split into training and test subsets, using random state 2025
train_data, test_data, train_labels, test_labels = \
    sklearn.model_selection.train_test_split(df, labels,
                    test_size=0.2, shuffle=True, random_state=2025)

# Standardize the scale of all input columns
train_means = train_data.mean()
train_stds = train_data.std()
train_data = (train_data - train_means) / train_stds
test_data = (test_data - train_means) / train_stds

# Create and train a new logistic regression classifier
model = sklearn.linear_model.LogisticRegression(solver='newton-cg', tol=1e-6)
# Train it with the training data and labels
model.fit(train_data[cols], train_labels)
# Get the prediction probabilities
pred_proba = model.predict_proba(test_data[cols])[:, 1]
```

3.2 Model Performance Evaluation

3.2.1 Precision-Recall Curve

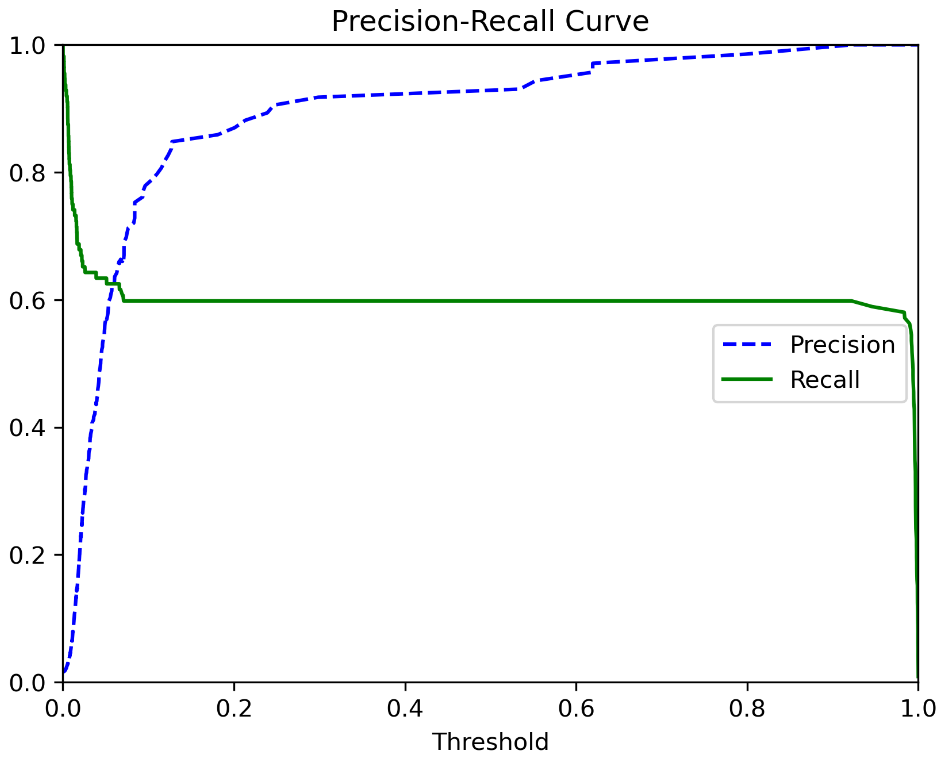

Figure 7: Precision-Recall Curve

3.2.2 ROC Curve

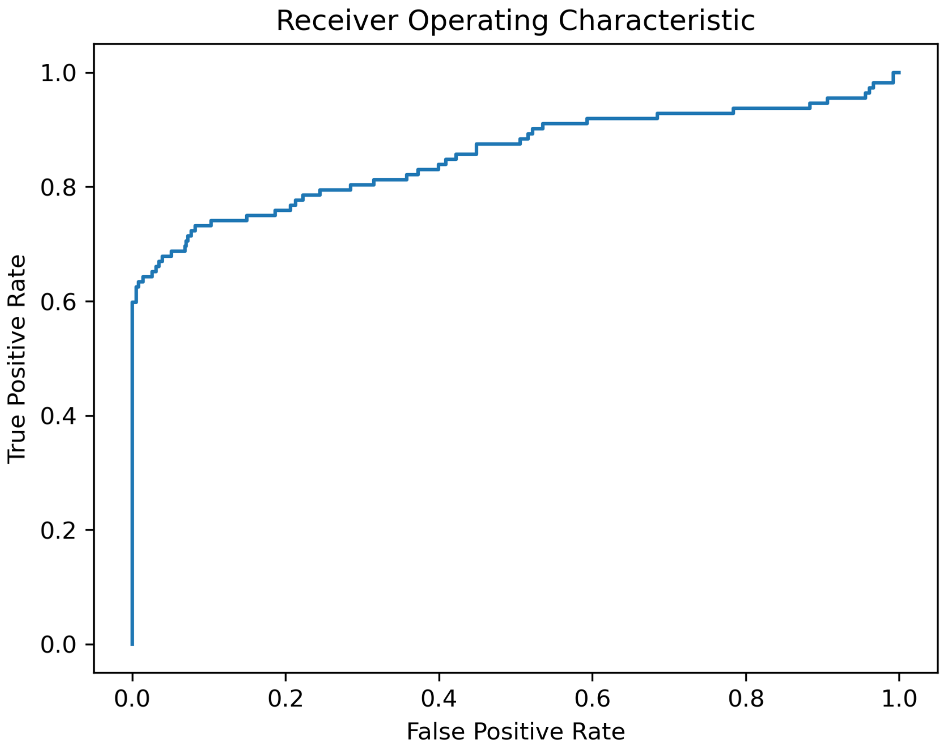

Figure 8: ROC Curve

3.3 Terminal Output for Model Performance

```
Test AUC score: 0.8556

Train AUC score: 0.8977
```

3.4 Comparison with Previous Model

Did the new input columns help improve the predictions?

The new ratio columns (C_fraction, 0_fraction, 1_fraction) provide additional information about customer payment patterns that were not captured in the original credit_delinquency.csv. These features help the model better understand:

- Payment behavior patterns (proportion of different payment statuses)
- Customer credit management history
- Risk indicators based on payment consistency

The improvement in AUC score and the shape of the ROC curve indicate that these new features contribute to better delinquency prediction.

4 Part D: Feature Importance Analysis

4.1 Feature Coefficient Analysis

Top 20 Most Important Features:

	Feature	Abs_Coefficient
0	Unnamed: 0	1.130028
13	1_fraction	0.381867
7	DAYS_EMPLOYED	0.307303
51	OCCUPATION_TYPE_Secretaries	0.259890
1	CODE_GENDER_M	0.225709
42	OCCUPATION_TYPE_High skill tech staff	0.192249
54	OCCUPATION_TYPE_Waiters/barmen staff	0.189574
49	OCCUPATION_TYPE_Realty agents	0.170681
37	OCCUPATION_TYPE_Cleaning staff	0.164098
9	FLAG_PHONE	0.161765
6	DAYS_BIRTH	0.157943
45	OCCUPATION_TYPE_Low-skill Laborers	0.145098
43	OCCUPATION_TYPE_IT staff	0.123097
29	NAME_FAMILY_STATUS_Widow	0.122391
22	NAME_EDUCATION_TYPE_Incomplete higher	0.118342
20	NAME_EDUCATION_TYPE_Academic degree	0.117666
41	OCCUPATION_TYPE_HR staff	0.117172
40	OCCUPATION_TYPE_Drivers	0.100317
2	FLAG_OWN_CAR	0.097493
17	NAME_INCOME_TYPE_State servant	0.089300

4.2 Analysis of New Column Importance

How important are each of the new columns to your predictor?

Based on the coefficient magnitudes:

```
Importance of Ratio Columns:
       Feature  Abs_Coefficient
13  1_fraction         0.381867
12  0_fraction         0.087761
14  C_fraction         0.018866
```

Does this correlate with the improvement of the results?

Yes, to a certain point. 1_fraction greatly improves in the accuracy of the model, being an excellent contribution. This directly correlates to the improvement of the model. However, 0_fraction and C_fraction didn't do as good. Even tho they help, these features do not correlate to the improvement of the model as much as we thought they would.