Technical Report: Assignment 3
Corinne Williams

Introduction

Working to predict whether firms in 2014 in the industry of 'Manufacturing computer, electronic and optical products', 6 different predictive classification models were used, each incorporating different amounts of the same predictor variables (some, including interaction terms) to compare each model's effectiveness in prediction. As such, the models used were (1) Logit Regression 1, (2) Logit Regression 2, (3) Logit Regression 3, (4) Logit Regression 4, (5) LASSO Logit Regression, (6) Random Forest Regression.

Data Cleaning and Preparation

The data is originally 287, 829 observations (46,412 firms throughout multiple years, from 2005-2016). It was narrowed down to 14,877 observations by filtering for the specified industry. The default variable (outcome variable, y) was then created by filtering for sales that were more than $0 in 2014, but did not exist otherwise when sales were 0 or missing values (N/A). Default was then created as a separate column/variable in the data dataframe if status_alive was equal to 1 (as a dummy variable). The outcome variable was set equal to that 'default' variable.

```
#creating y variable ('default')
# generate status_alive; if sales larger than zero and not-NA, then firm is alive
data["status_alive"] = (data["sales"] > 0 & (False == data["sales"].isna())).astype(int)

# defaults in 1 year if there are sales in this year but no sales one year later
# Status_in_one_year: data.groupby('comp_id')['status_alive'].shift(-1)
data["default"] = (
    (data["status_alive"] == 1)
    & (data.groupby("comp_id")["status_alive"].shift(-1) == 0)
).astype(int)

#setting y equal to 'default'
y = data['default']
```

Next, the dataset variables were cleaned, imputed, and flags were created for imputed and missing variables. The predictor variables were as such: amortization, current assets, current liabilities, extra expenditures, extra income, fixed assets, income before tax, intangible assets, inventories, liquid assets, material expenses, personnel expenses, profit loss year, share equity, subscribed capital, tangible assets, ceo count, foreign ceo percentage, percentage of female ceos, days a ceo was in-office, whether a firm was female-only, whether a firm was male-only, whether a firm was mixed-gender, whether the ceo is domestic, whether the ceo is foreign, whether the ceo(s) are of mixed origin (domestic and foreign), whether the headquarters are located in a capital city, whether the headquarters are located in a big city, whether the headquarters are located in another type of city, weather the headquarters are located in the Central United States, whether the headquarters are located in the East United States, and whether the headquarters are located in the West United States. Additionally, category variables were turned into dummies. Because most of the integer variable distributions were skewed, the missing values were imputed with the median values. The categorical variable observations with missing values were imputed using the mode due to the qualitative nature of the observations.

```
#imputing missing amort values with median (less affected by outliers)
#creating flag_variable for missing values and imputed values
data.amort.isnull().sum()
data['amort_missing_flag'] = data.amort.isnull().astype(int)
data['amort_imputed_flag'] = data.amort.isnull().astype(int)
data.fillna({'amort':data['amort'].median()}, inplace = True)

#imputing missing curr_assets values with median
#creating flag_variable for missing values and imputed values
data.curr_assets.isnull().sum()
data['curr_assets_missing_flag'] = data.curr_assets.isnull().astype(int)
data['curr_assets_imputed_flag'] = data.curr_assets.isnull().astype(int)
data.fillna({'curr_assets':data['curr_assets'].median()}, inplace = True)
```

Example of imputation and flag variable creation seen above. A list of all of the variables was created and the reference groups for each were listed:

```
variables = ['amort', 'sales', 'curr_assets', 'curr_liab', 'extra_exp', 'extra_inc', 'fixed_assets', 'inc_bef_tax',
             'intang_assets', 'inventories','liq_assets', 'material_exp', 'personnel_exp', 'profit_loss_year',
             'share_eq', 'subscribed_cap', 'tang_assets', 'ceo_count', 'foreign_ceo', 'female',
             'inoffice_days', 'female_only', 'male_only', 'domestic', 'not_domestic', 'capital_city','other_hq_city'
             'east', 'west']
#reference group for gender is mixed_gender firms
#refrence group for origin is mix CEO origin
#reference group for urban_m is big_city HQ location
#reference group for region_m is Central HQ region
```

Creating Holdout and Training Samples:

The holdout set was pre-defined as companies in the selected industry that are small or medium enterprises, in which the yearly sales in 2014 were between 1000 EUR and 10 million EUR. The training sample was then all other observations that did not include those in the holdout set. The holdout sample ended up including 1,037 firms.

```
#criteria for the holdout sample so I can mask the training sample with !holdout_criteria
holdout_criteria = (data['year'] == 2014) & ((data['sales'] >=1000) & (data['sales']<=10000000))
```

```
#holdout sample creation
holdout = data[holdout_criteria]
print(f'Holdout shape: {holdout.shape}')
holdout.head()
```

The training sample was simply created from the criteria that was not included in the holdout. It ended up consisting of 13,252 firms.

```
#creating training sample
training = data[~holdout_criteria]
training = training.loc[training['year'] != 2016]
print(f'Training shape: {training.shape}')
training.head()
```

```
Training shape: (13252, 98)
```

Because the logit regressions would not accept null values, the null values in the sales variable were imputed as zeroes.

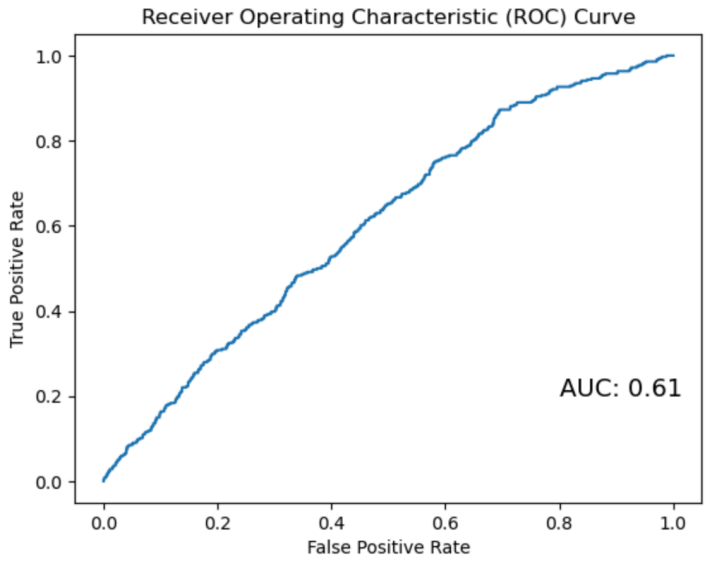
```
#Logit regression won't accept null values in sales, so imputing to 0
#imputing missing sales values with 0
#creating flag_variable for missing values and imputed values
training.sales.isnull().sum()
training['sales_missing_flag'] = training.sales.isnull().astype(int)
training['sales_imputed_flag'] = training.sales.isnull().astype(int)
training['sales'].fillna(0, inplace=True)
```

Model 1: Logit Regression

Model 1 used all the variables along with some interaction terms created.

```
##interactions

training['interaction_curr_assets_vs_curr_liab'] = training['curr_assets'] * training['curr_liab']
training['interaction_inoffice_days_vs_foreign_ceo'] = training['inoffice_days'] * training['foreign_ceo']
training['interaction_tang_vs_liq_assets'] = training['tang_assets'] * training['liq_assets']
training['interaction_female_vs_male'] = training['female_only']  * training['male_only']
training['interaction_foreign_ceo_vs_ceo_count'] = training['foreign_ceo']  * training['ceo_count']
training['interaction_capital_city_vs_not_domestic'] = training['capital_city']  * training['not_domestic']
training['interaction_profit_loss_year_vs_extra_exp'] = training['profit_loss_year']  * training['extra_exp']


interaction_terms = ['interaction_curr_assets_vs_curr_liab', 'interaction_inoffice_days_vs_foreign_ceo', 'interactio
```

The scale for the X1 variables had to be changed or else the coefficients would read as 0 for each variable. y became training['default'] the model was cross-validated with 5 folds. Using a defined function, the CV RMSE was calculated and came out to be 0.164. Prediction measures were calculated such as precision, recall, f-1, accuracy, as well as the AUC-ROC score which turned out to be 0.60- meaning that the model has some discriminatory power to be able to 'discriminate' between firms that defaulted and firms that did not. Next, the ROC was visualized along with the AUC printed as well as the optimal threshold used to classify the predictions.

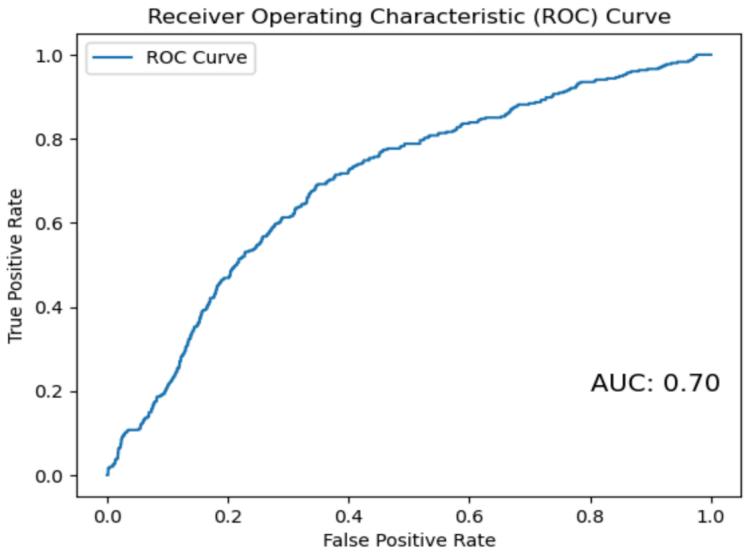



```
Optimal Threshold: 0.026666474175300173
```

Model 2: Logit Regression

Model 2 proved to be the best logit regression using only the first 9 predictor variables with no interaction terms. It was also cross-validated on 5 folds and the CV RMSE came out to 0.258. Prediction measures were calculated such as precision, recall, f-1, accuracy, as well as the AUC-ROC score which turned out to be 0.70, meaning that the model has pretty acceptable discriminatory power (better than Model 1). The ROC was visualized along with the AUC printed as well as the optimal threshold used to classify the predictions.

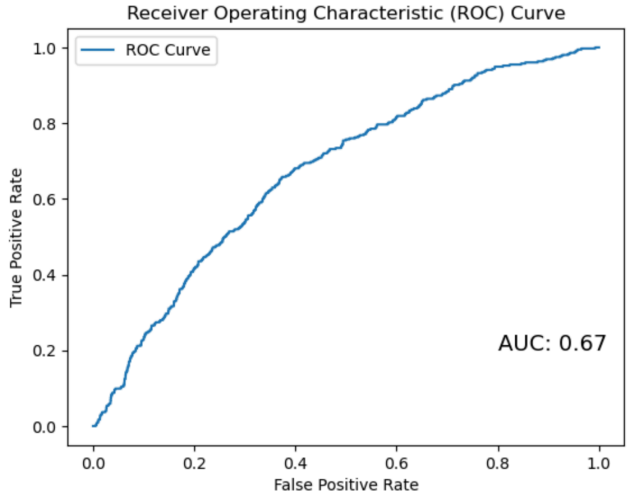
```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96     12898
           1       0.06      0.11      0.08       354

    accuracy                           0.93     13252
   macro avg       0.52      0.53      0.52     13252
weighted avg       0.95      0.93      0.94     13252

AUC-ROC Score: 0.6971154814875166
```




```
Optimal Threshold: 0.45
```

Model 3: Logit Regression

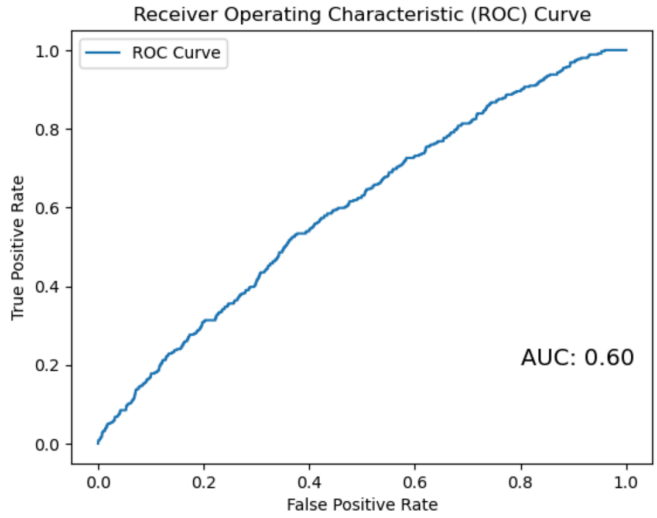
Model 3 used the first 18 predictor variables with no interaction terms. It was also cross-validated on 5 folds and the CV RMSE came out to 0.258. Prediction measures were calculated such as precision, recall, f-1, accuracy, as well as the AUC-ROC score which turned out to be 0.67, meaning that the model has some discriminatory power (better than Model 1, worse than Model 2). The ROC was visualized along with the AUC printed as well as the optimal threshold used to classify the predictions.




```
Optimal Threshold: 0.40
```

Model 4: Logit Regression

Model 4 used all predictor variables with no interaction terms. It was also cross-validated on 5 folds and the CV RMSE came out to 0.165. Prediction measures were calculated such as precision, recall, f-1, accuracy, as well as the AUC-ROC score which turned out to be 0.60, , meaning that while the model has some discriminatory power, it has the weakest AUC-ROC values out of all the models so far. The ROC was visualized along with the AUC printed as well as the optimal threshold used to classify the predictions.




```
Optimal Threshold: 0.07
```

Model 5: LASSO Logit Regression

Model 5 used LASSO selected features to create the logit regression. It was also cross-validated with 5 folds and the CV RMSE came out to be 0.165. Prediction measures were calculated such as precision, recall, f-1, accuracy, as well as the AUC-ROC score which turned out to be 0.63, meaning that while the model has some discriminatory power, it is still pretty weak. The ROC was visualized along with the AUC printed as well as the optimal threshold used to classify the predictions.

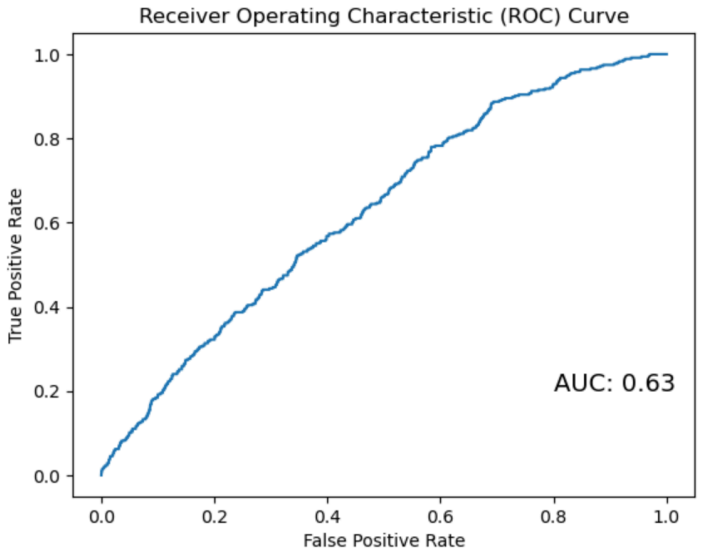
```
#changed penalization to 0.1 because it was too high (selected features returned no features)
lasso_model = LogisticRegression(penalty='l1', C=0.1, solver='liblinear')
lasso_model.fit(X_lasso_scaled, y)
```

```
LogisticRegression
LogisticRegression(C=0.1, penalty='l1', solver='liblinear')
```

```
#selected features chosen by LASSO
selected_features = X1.columns[lasso_model.coef_[0] != 0]
print("Selected Features:", selected_features)
#15 selected_features/interactions
```

```
Selected Features: Index(['curr_liab', 'extra_exp', 'inventories', 'share_eq', 'subscribed_cap',
       'tang_assets', 'ceo_count', 'foreign_ceo', 'inoffice_days',
       'female_only', 'male_only', 'east', 'west',
       'interaction_inoffice_days_vs_foreign_ceo',
       'interaction_profit_loss_year_vs_extra_exp'],
      dtype='object')
```

```
#using selected features selected by lasso
lasso_selected_features = ['curr_liab', 'extra_exp', 'inventories', 'share_eq', 'subscribed_cap',
       'tang_assets', 'ceo_count', 'foreign_ceo', 'inoffice_days',
       'female_only', 'male_only', 'east', 'west',
       'interaction_inoffice_days_vs_foreign_ceo',
       'interaction_profit_loss_year_vs_extra_exp']

X6 = training[lasso_selected_features]
```

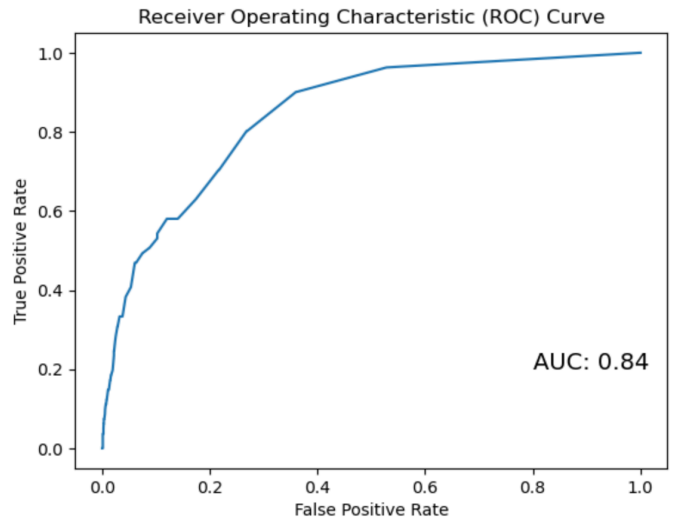



```
Optimal Threshold: 0.02555382509788696
```

Model 6: Random Forest

Model 6 used the features from model 3's Logit Regression due to it providing the best AUC-ROC score for the random forest model. X train and test as well as Y train and test samples were created in order to train the model on a test sample. The model was also cross-validated with 5 folds and the CV RMSE came out to be 0.164. The AUC-ROC score was the highest out of any of the models at 0.84 meaning that the model has pretty excellent discriminatory power. Model 6 had the best AUC-ROC and relatively one of the lowest expected loss values.

```
#scaling X3 (gives best RF AUC)
X_rf_scaled = scaler.fit_transform(X3)

#data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_rf_scaled, y, test_size=0.2, random_state=42)

#rf classifier
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)

#fitting model on training data
random_forest_model.fit(X_train, y_train)

#making predictions on test set
y_pred_rf = random_forest_model.predict(X_test)

#cv rmse
cv_rmse6 = cross_val_score(random_forest_model, X_rf_scaled, y, cv=5, scoring=make_scorer(rmse_scorer))
print("Cross-Validated RMSE:", np.mean(cv_rmse6))

#evaluate model
print("Accuracy Score:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
print("AUC-ROC Score:", roc_auc_score(y_test, random_forest_model.predict_proba(X_test)[:, 1]))
```




```
Optimal Threshold: 0.02
```

Model Comparison:

Using predicted probabilities and then binary classifications, expected losses for each model were found using a function to define expected loss.

```python
#defining a loss function (would be from 0 to 1), 0 being the best
def zero_one_loss(y_true, y_pred):
    return 1 - accuracy_score(y_true, y_pred)

#predicted probabilities for each model
y_prob = logistic_model.predict_proba(X1_scaled)[:, 1]
y_prob2 = logistic_model2.predict_proba(X2)[:, 1]
y_prob3 = logistic_model3.predict_proba(X3)[:, 1]
y_prob4 = logistic_model4.predict_proba(X4)[:, 1]
y_prob5 = lasso_model.predict_proba(X6)[:, 1]
y_prob_rf = random_forest_model.predict_proba(X_test)[:, 1]



#using each model's optimal threshold to get binary predictions
y_pred = (y_prob >= optimal_threshold).astype(int)
y_pred2 = (y_prob2 >= optimal_threshold2).astype(int)
y_pred3 = (y_prob3 >= optimal_threshold3).astype(int)
y_pred4 = (y_prob4 >= optimal_threshold4).astype(int)
y_pred5 = (y_prob5 >= optimal_threshold5).astype(int)
y_pred_rf = (y_prob_rf >= optimal_threshold_rf).astype(int)

#getting expected loss for each model
loss = zero_one_loss(y, y_pred)
loss2 = zero_one_loss(y, y_pred2)
loss3 = zero_one_loss(y, y_pred3)
loss4 = zero_one_loss(y, y_pred4)
loss5 = zero_one_loss(y, y_pred5)
loss_rf = zero_one_loss(y_test, y_pred_rf)
```

A data frame was then created to show the comparison between each model, along with the metrics of: # of predictors, CV RMSE, CV AUC, Optimal Threshold, and CV Expected Loss.

	Models	# of Predictors	CV RMSE	CV AUC	Optimal Threshold	CV Expected Loss
0	M1	36	0.164350	0.602	0.026666	0.680350
1	M2	9	0.258435	0.696	0.449890	0.345080
2	M3	18	0.258443	0.679	0.404174	0.372925
3	M4	29	0.165728	0.599	0.066641	0.375868
4	LASSO	15	0.165272	0.626	0.025554	0.573800
5	RF	18	0.164587	0.844	0.020000	0.352320

Overall Model 6 (Random Forest) was found to be the best model (relatively low expected loss and very high AUC-ROC score in comparison to the other models).

Prediction with Random Forest on Holdout Sample:
X and y variables for the holdout sample were created. Then the random forest model was used to predict the y values for the holdout sample (whether a company would default or not)- using the same optimal threshold from the random forest training sample model. The number of companies predicted to default was 276/1,037.

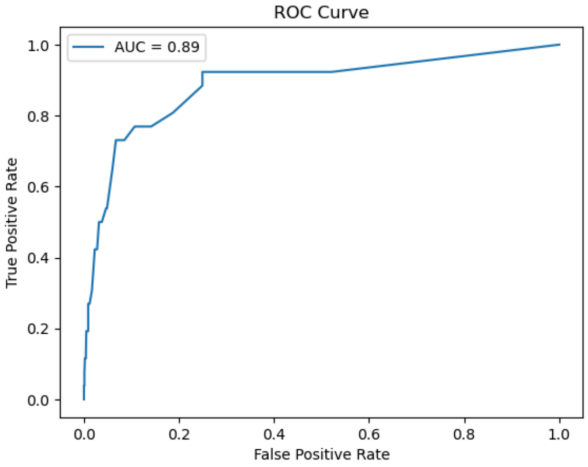
```
#holdout X and y variables
X_holdout = holdout[X3.columns]
y_holdout = holdout['default']
```

```
#scale x
X_holdout_scaled = scaler.transform(X_holdout)

#predicting holdout set
y_prob_rf_holdout = random_forest_model.predict_proba(X_holdout_scaled)[:, 1]

#optimal threshold
fpr_rf_holdout, tpr_rf_holdout, thresholds_rf_holdout = roc_curve(y_holdout, y_prob_rf_holdout
roc_auc_rf_holdout = roc_auc_score(y_holdout, y_prob_rf_holdout)

optimal_threshold_rf_holdout = thresholds_rf_holdout[np.argmax(tpr_rf_holdout - fpr_rf_holdout

#binary predictions using optimal threshold
y_pred_rf_holdout = (y_prob_rf_holdout >= optimal_threshold_rf_holdout).astype(int)

#new columnn for those predicted to default
holdout['predicted_default'] = y_pred_rf_holdout
```

```
#num of companies predicted to default
num_predicted_default = y_pred_rf_holdout.sum()

print(f"Number of companies predicted to default: {num_predicted_default}")
#At first said 161 firms, then reran and it changed to 276 firms predicted to default???
```

```
Number of companies predicted to default: 276
```

The assignment told us in the beginning that 56 firms out of 1,037 should have defaulted; yet my random forest model predicted 276 out of 1,037 firms would default; despite a very high AUC-ROC score of 0.89. In the end, descriptive statistics were calculated for the prediction and the holdout set in order to provide a benchmark of comparison for the classmates' results.




```
Optimal Threshold: 0.02333333333333333
```

```
Brier Score (RF): 0.0211
AUC (RF): 0.8861
Optimal Threshold (RF): 0.0233
Accuracy (RF): 0.7551
Sensitivity (RF): 0.9231
Specificity (RF): 0.7507
Expected Loss (RF): 0.7580

Descriptive Statistics for the entire holdout set:
Number of Firms: 1037
Number of Defaulted Firms: 276
Number of Firms that Stayed Alive: 761
Mean Sales: 490202.2179
Min Sales: 1070.3704
Max Sales: 9576485.0000
```

Conclusion:

Despite the Random Forest model proving to be the best, its prediction was far off from what was expected. When given the assignment, it was stated that 56 firms should default; however, model 6 predicted 276. This large difference can also be seen in the change in metrics from the Random Forest training model to the Random Forest model used to predict the holdout sample. For example, in Model 6, the expected loss value was 0.352; however, when predicting the holdout sample, that value jumped to 0.758. Interestingly enough, the AUC-ROC score also increased to 0.89 despite the biased prediction. I suspect this is due to the large bias in the holdout sample classes. There is a large difference between the minority and majority classes. Therefore, I feel that perhaps the training sample is not a good representation of the holdout, leading to a pretty skewed prediction.