STA 4634 Final Report

By: Charlie Diaz and Tyler Boshaw

I. Problem Statement

We will be creating and testing a loan approval prediction model utilizing different model types and several characteristics from a range of predictors such as age, income, and loan intent. This project will attempt to predict the probability that a loan is approved and analyze the factors that most heavily contribute to the loan approval process. We found this interesting because it has real world applications to both banks and those who are applying for loans.

II. Data

The dataset we used for this project is from a Kaggle competition in which the goal was to predict whether an applicant was approved for a loan. The dataset was artificially generated based off of a real loan approval dataset. Kaggle uses artificially generated data in many of its competitions so that it is not possible to find the target data anywhere and cheat the competition. Because this data was artificially generated, there are no missing values and therefore no imputation was needed. However, we did need to deal with the categorical data. For this we used one hot encoding since our categorical variables had no inherent ordering. In Fig. 1 below, we have a correlation matrix heatmap with the categorical features dropped.

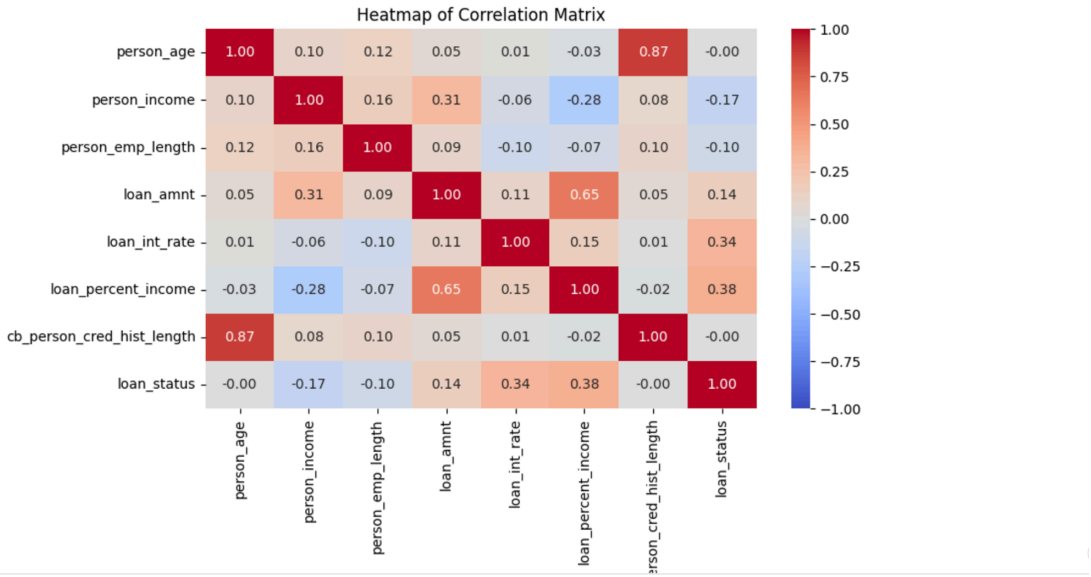

Fig. 1: Correlation Matrix

It can be seen that "person_age" and "cb_person_cred_hist_length" have a high correlation with each other but no correlation with our target, "loan status", so we will test how dropping these features affects the model.

III. Method and Results

We began by creating and testing a logistic regression model for our dataset. We transformed any categorical data into numerical data and made sure to drop the target from our predictors. Upon the creation of the logistic regression model, we found that all predictors were statistically relevant, allowing us to test each model with all the data available. For the logistic regression metrics, we tested the model using a confusion matrix which is below in Fig. 2, and found an accuracy of 94.68%, which is very good for a start.

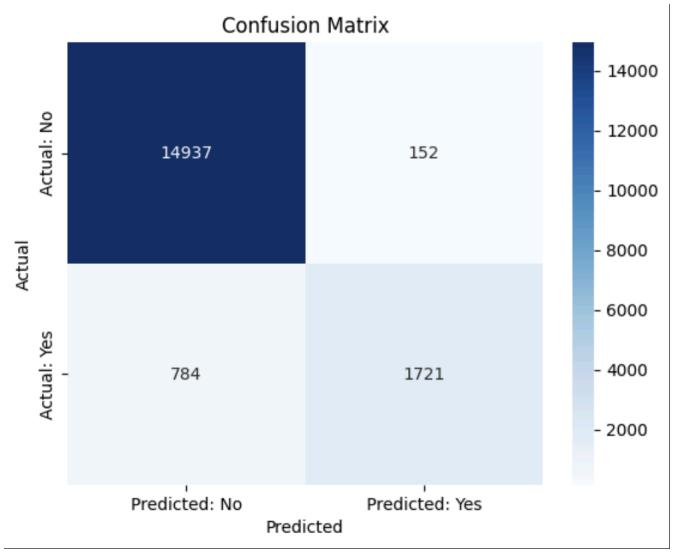

Fig.2 : Confusion Matrix of Logistic Regression Model

We then included an F1 Score for the logistic model, which gave us a 0.79 accuracy rating which is good but not great. Next, we found an ROC score of 0.8773 for the model. Our ROC curve can be seen in Fig. 3 below. Overall, there is definitely room for improvement to squeeze out from the classification tree models.

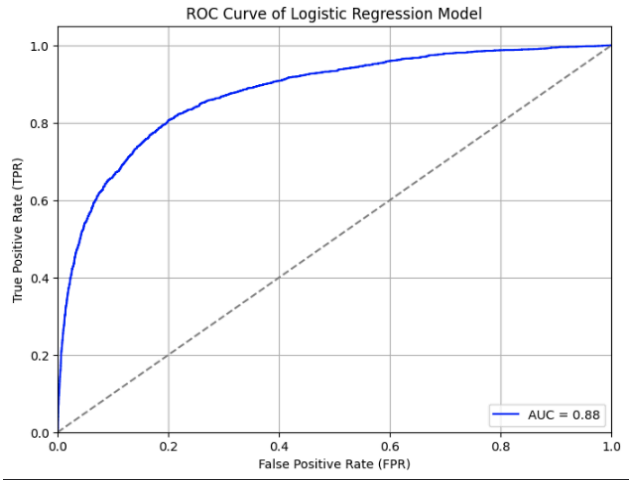

Fig.3 : ROC Curve for Logistic Regression

Moving into our other models, we decided to look at a classification tree model, which we then added ensemble methods of bagging, XGBoost, and random forests onto. For the main classification tree model, we found the best accuracy just shy of 95%, which was achieved with a tree depth of 7 nodes. After about 7-8 nodes, the model begins to struggle and then the accuracy begins to drop as seen in Fig. 4 below.

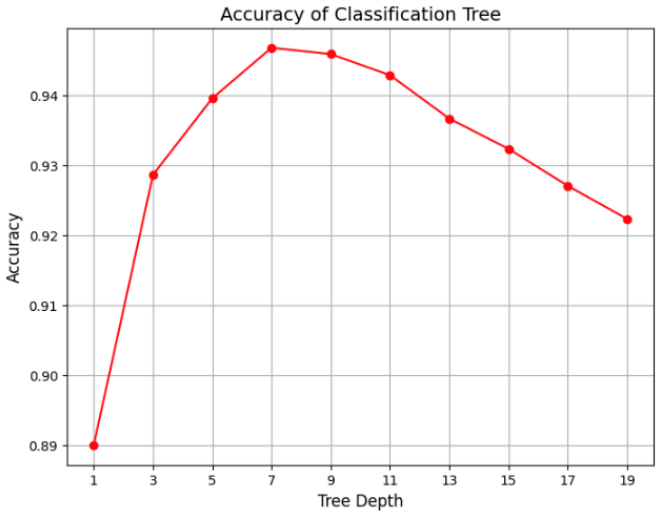

Fig. 4: Classification Tree Model Accuracy

Moving onto the first ensemble method, we looked into the bagging method for our model. We tested the model by creating a bootstrapping sample set of 25, which resulted in a top accuracy of 94.8%, which is extremely accurate. For a reasonable model, we could have stopped closer to 10 samples, but for testing we used 25. You can see this in Fig. 5 below which shows our accuracy over the sample sets.

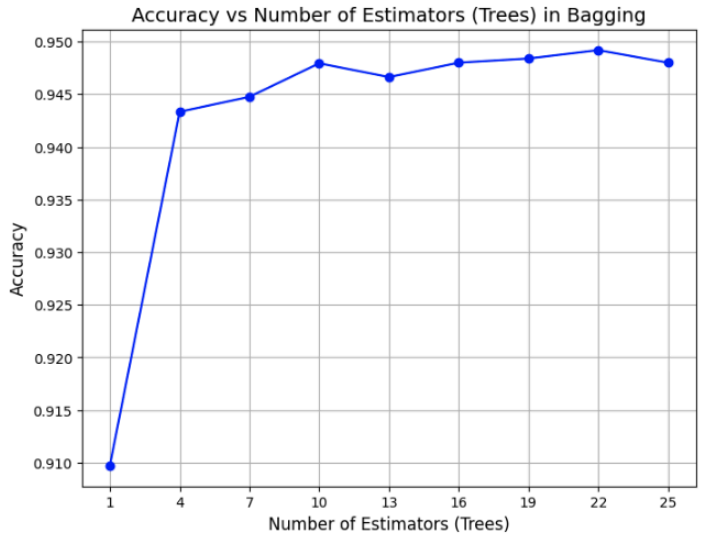

Fig. 5: Bagging Method Accuracy

We also included a boosting ensemble method, utilizing specifically XGBClassifier which is a very popular technique in the machine learning toolset.

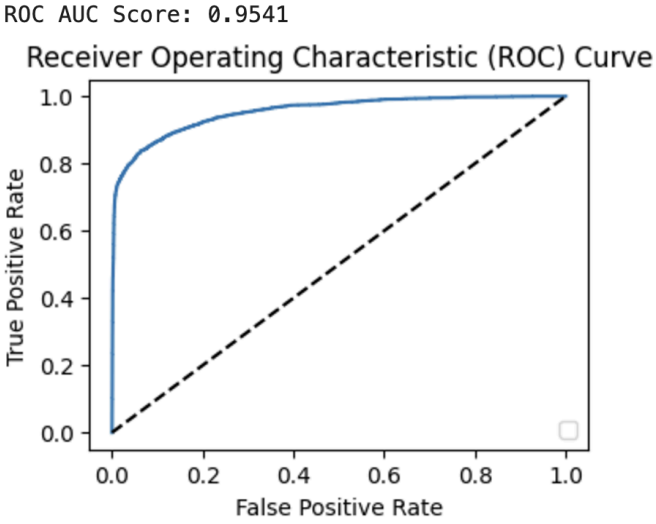

Fig. 6: XGBClassifier ROC curve

For our last method, we looked at the randomForests ensemble method. To our surprise, the difference between the bagging and randomForests methods was minimal, but it seems that the randomForests is a more stable model for prediction. As seen below in Fig. 7, we also utilized 25 sample sets like the bagging method, resulting in an accuracy of 94.8%.

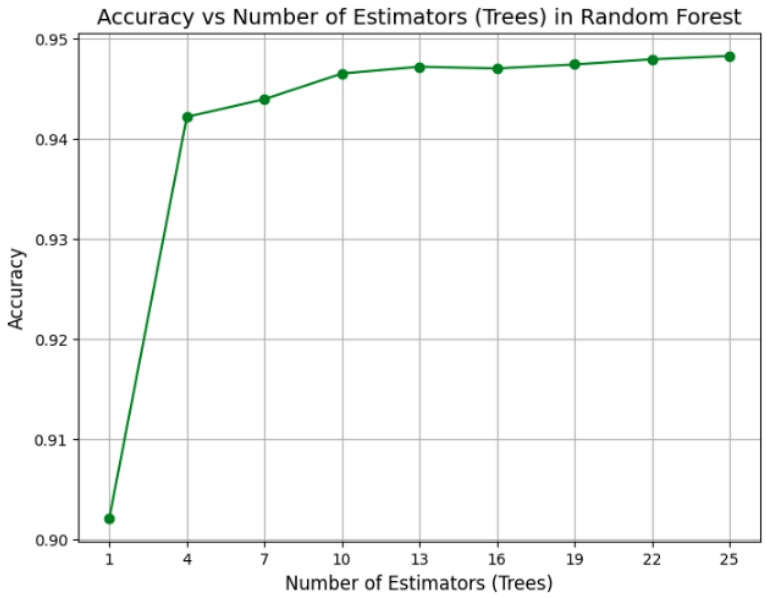

Fig. 7: RandomForest Model Accuracy

IV. Discussion

For this dataset and prediction problem, XGBClassifier provided the optimal fit. This was expected because the XGBClassifier combines outputs of decision trees to enhance learning and create a robust model. Many of the top scores in the kaggle competition for this dataset also used some form of gradient boosting, proving its efficacy for classification problems such as the one we were predicting. As discussed in the data section the predictors “person_age” and “cb_person_cred_hist_length” provided no additional prediction value to our models and in some cases dropping them actually improved the quality of our models. Overall, we managed to research several models, giving us valuable insight into which models would provide us with a clear and accurate loan prediction tool.

V. Code and Output

Correlation Matrix

```
train_nocat = train.select_dtypes(exclude=['object', 'category'])

plt.figure(figsize=(9, 5))
sns.heatmap(train_nocat.corr(), annot=True, fmt=".2f", cmap="coolwarm", cbar=True, vmax=1, vmin=-1)

plt.title("Heatmap of Correlation Matrix")
plt.show()
```

Logistic Regression Model

```
# [ logistic regression model ]
# transforming data types from categorial to values
data = train.loc[:,train.dtypes=="object"].columns
train[data] = train[data].apply(lambda x: LabelEncoder().fit_transform(x))

X = train.drop(target, axis=1)
Y = train[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.3, random_state=123, stratify=Y
)

# fitting basic logistic model
logr = sm.Logit(y_train, X_train)
results = logr.fit()
sum = results.summary()
print(sum)
```

```
Optimization terminated successfully.
         Current function value: 0.274160
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:            loan_status   No. Observations:                41051
Model:                          Logit   Df Residuals:                    41040
Method:                           MLE   Df Model:                           10
Date:                Sun, 08 Dec 2024   Pseudo R-squ.:                 0.3301
Time:                        19:21:00   Log-Likelihood:                -11255.
converged:                       True   LL-Null:                       -16801.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
person_age                      -0.0637      0.005    -12.303      0.000      -0.074      -0.054
person_income                -2.453e-05   1.65e-06    -14.821      0.000   -2.78e-05   -2.13e-05
person_home_ownership            0.2880      0.015     19.604      0.000       0.259       0.317
person_emp_length               -0.0230      0.005     -4.498      0.000      -0.033      -0.013
loan_intent                     -0.1581      0.010    -15.554      0.000      -0.178      -0.138
loan_grade                       1.7728      0.040     44.414      0.000       1.695       1.851
loan_amnt                     2.133e-05   8.05e-06      2.650      0.008    5.56e-06    3.71e-05
loan_int_rate                   -0.3057      0.013    -23.297      0.000      -0.331      -0.280
loan_percent_income              8.1062      0.406     19.941      0.000       7.309       8.903
cb_person_default_on_file       -0.1283      0.046     -2.799      0.005      -0.218      -0.038
cb_person_cred_hist_length       0.0879      0.008     10.883      0.000       0.072       0.104
================================================================================================
```

Logistic Regression Confusion Matrix / F1 Score

```python
## confusion matrix
conf = confusion_matrix(y_test, y_pred)

# heatmap of confusion matrix
sns.heatmap(conf, annot=True, fmt="d", cmap="Blues", xticklabels=["Predicted: No", "Predicted: Yes"], yticklabels=["Actual: No", "Actual: Yes"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()

# accuracy metrics of confusion matrix
conf_acc = ((conf[0,0]+conf[1,1]) / (conf[0,0]+conf[0,1]+conf[1,0]+conf[1,1]))
print(f"Confusion Matrix Accuracy: {conf_acc:.2%}\n")

# F1 score
precision = conf[1,1] / (conf[1,1] + conf[0,1])
recall = conf[1,1] / (conf[1,1] + conf[1,0])
f1_score = 2 * ((precision * recall) / (precision + recall))
print(f"F1 Score: {f1_score:.2}")
```

```
Confusion Matrix Accuracy: 94.68%

F1 Score: 0.79
```

ROC Curve for Logistic Regression Model

```python
# fit the logistic regression model
logr = sm.GLM(y_train, X_train, family=sm.families.Binomial())
results = logr.fit()

# get predicted probabilities for the positive class
predictions_prob = results.predict(X_test)

# compute the ROC curve
fpr, tpr, thresholds = roc_curve(y_test, predictions_prob)
roc_auc = auc(fpr, tpr)
```

Classification Tree Model

```
# main classification-tree model (depth=7 for best accuracy according to loop below)
dt_classifier = DecisionTreeClassifier(random_state=123, max_depth=7)
dt_classifier.fit(X_train, y_train)

# makes predictions and evaluates the model
y_pred = dt_classifier.predict(X_test)
print(f"Decision Tree Accuracy: {accuracy_score(y_test, y_pred):.3f}")

# accuracy measurement of classification tree
depths = range(1, 20, 2)  # Try depths from 1 to 20
accuracies = []

for depth in depths:
    # initialize the classifier with different max_depth values
    dt_classifier = DecisionTreeClassifier(random_state=123, max_depth=depth)
    dt_classifier.fit(X_train, y_train)

    # predict and calculate accuracy
    y_pred = dt_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
```

Bagging Model

```
# bagging model
bagging_classifier = BaggingClassifier(estimator=DecisionTreeClassifier(random_state=123),
                                       n_estimators=25, random_state=123)

bagging_classifier.fit(X_train, y_train)

# evaluate the bagging model
y_pred_bagging = bagging_classifier.predict(X_test)
print(f"Bagging Accuracy: {accuracy_score(y_test, y_pred_bagging):.3f}")

# accuracy measurement of classification tree
n_estimators_range = range(1, 26, 3)
accuracies_bagging = []

for n_estimators in n_estimators_range:
    # create a new bagging model
    bagging_classifier = BaggingClassifier(estimator=DecisionTreeClassifier(random_state=123),
                                           n_estimators=n_estimators, random_state=123)
    bagging_classifier.fit(X_train, y_train)

    # predict and calculate accuracy
    y_pred_bagging = bagging_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred_bagging)
    accuracies_bagging.append(accuracy)
```

XGBClassifier

```
model = XGBClassifier(eval_metric='logloss', tree_method='hist', verbosity=0)
model.fit(X_train, y_train)
```

```
stratified_cv = StratifiedKFold(n_splits=5)
cv_scores = cross_val_score(model, X, y, cv=stratified_cv, scoring='roc_auc')
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())
```

```
Cross-validation scores: [0.95116594 0.9540867  0.9527728  0.95815071 0.95306137]
Mean cross-validation score: 0.9538475033232074
```

RandomForests Model

```python
# randongForest model
rf_classifier = RandomForestClassifier(n_estimators=25, random_state=123)
rf_classifier.fit(X_train, y_train)

# evaluate the randomForest model
y_pred_rf = rf_classifier.predict(X_test)
print(f"Random Forest Accuracy: {accuracy_score(y_test, y_pred_rf):.3f}")

# accuracy of randomForest model
n_estimators_range = range(1, 26, 3)
accuracies_rf = []

for n_estimators in n_estimators_range:
    # create a new randomForest model with a different number of trees
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators, random_state=123)
    rf_classifier.fit(X_train, y_train)

    # predict and calculate accuracy
    y_pred_rf = rf_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred_rf)
    accuracies_rf.append(accuracy)
```

If you would like to see all of our code, you can see it here. There was way too much to put in just the project alone.
https://www.kaggle.com/code/charliediaziii/project-sta4634

VI. References

Walter Reade and Ashley Chow. Loan Approval Prediction. https://kaggle.com/competitions/playground-series-s4e10, 2024. Kaggle.