ASNA Hackathon:
Predicting Automotive Claims That Exceed $1000

Team: *ASNA Chefs*

November 1st, 2024

Table of Contents

Executive Summary 2

Problem Statement and Assumptions 3

The Model

1. Random Forest 4
2. Feature Selection 4
3. Optimizing our Model 5
4. The Model 6

Results and Discussion 6

Potential Applications 7

Conclusion 7

Code Appendix 8

Executive Summary

To Whom It May Concern,

In today's insurance landscape, predictive modeling is critical for assessing risk and optimizing claims processes. We developed a machine learning model specifically to predict automotive insurance claims over $1000, enabling insurers to make data-driven decisions and gain insights across actuarial applications. By employing an intuitive framework, our model achieves over 92% accuracy, with features designed to make the model's outputs accessible and understandable for both technical and non-technical partners.

We used a Random Forest model to assess feature importance and identify the key factors influencing claims outcomes. Recursive Feature Elimination (RFE) was then implemented to determine the six features with the strongest predictive power: Customer Lifetime Value, Income, Marital Status Index, Number of Policies, Days Since Effective Date, and State. These features were selected based on their impact on model accuracy, using machine learning methods, forming the core of our predictive framework.

Through careful tuning of model parameters using Grid Search, we optimized the model to recognize and learn from complex patterns in claims data. Adjusting various settings, such as how heavily a feature is relied on and the number of times to train the model without overfitting was a crucial step in ensuring our model would be optimized for claims prediction. This approach allowed the model to correctly predict claims over $1000 with a precision of 92.8%. Additionally, the model demonstrated over 98% accuracy in predicting claims under $1000, highlighting its effectiveness and potential to support reliable, high-stakes decision-making.

Our model equips business partners with a powerful tool for assessing claims risk, enhancing actuarial precision, and informing strategic decisions. These predictive insights can aid in claim evaluations and improve financial forecasting, therefore contributing to a more effective, data-driven approach to risk management. By integrating this model into the claims process, we anticipate aiding business partners through more reliable predictions for claims over $1000. This would support a balanced risk approach and align with organizational goals for improved accuracy and efficiency.

Problem Statement

Develop a model to predict auto insurance claims that exceed $1000 by analyzing how different factors impact claims.

Assumptions

1. **Future trends will be similar to historical data**
   Justification: It is challenging to predict how non-human events, such as weather and technological malfunctions, especially with the rise of climate change, might impact the influence of the given features on claims. Thus, to ensure the validity of our model, we will assume this data accurately represents trends for the coming years.

2. **Data completeness and accuracy.**
   Justification: The model operates under the assumption that the dataset used is complete, up to date, and accurately reflects the population it represents. Significant biases, fraudulent claims, or missing data that could skew results, altering the model's accuracy and consistency. We assume that none exists in the dataset.

3. **Data uniqueness**
   Justification: When creating a predictive model, having features that overlap with each other can severely hinder its ability to make accurate predictions. Having multiple features with similar or conflicting data can cause for the model to fail to generalize new, unseen data. This can decrease the model's ability to learn effectively from the training data.

The Model

1. **Random Forest**[1]

   The model uses a tree-like structure where it sorts an observation into two groups by asking questions about the features (ex. Income) that gets more and more specific until the observation has been sorted into claims above $1000 and claims at or below $1000.

   Random Forest combines multiple of these tree-like structures to create a forest of trees, which work together to make more accurate and stable predictions than any single decision tree alone. The model then builds several decision trees from the same training dataset, but each tree is trained on a random subset of the data. The forest takes a majority vote of the predictions from all trees based on how accurate each tree is and whichever class accumulates the most votes becomes the final prediction.

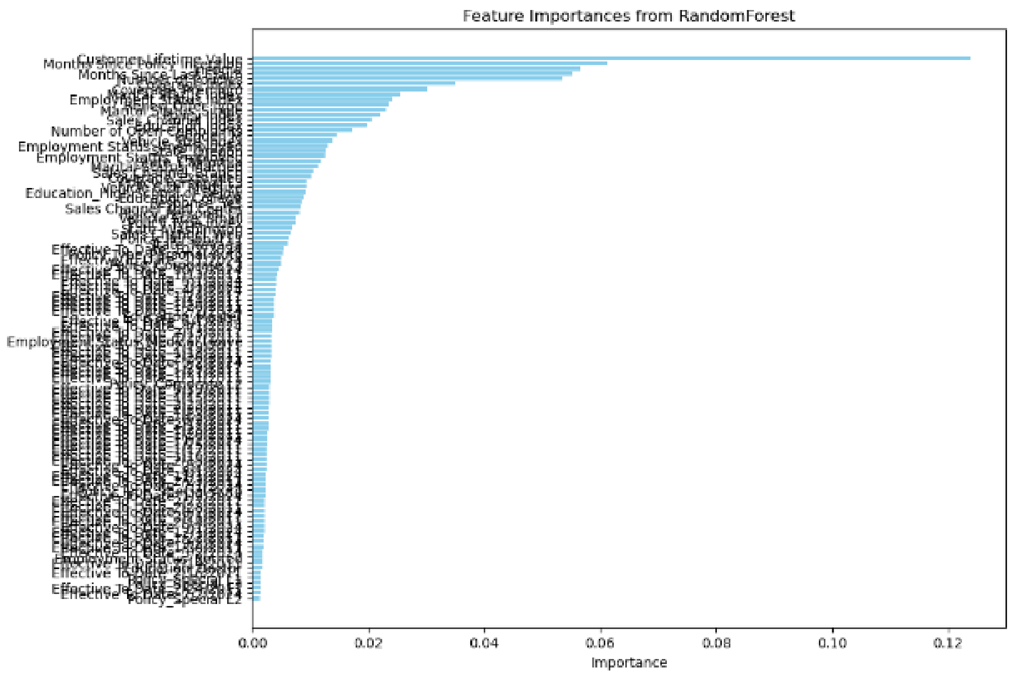

Figure 1: Random Forest Classification for every feature

2. **Feature Selection**[2]

   With a way to rank the importance of each feature, we now need to find a way to choose which features to train our model on. As highlighted in Figure 1, there are a plethora of features contained in this dataset, with many having little to no importance on the final model. With the broad range of attributes related to customer profiles and

[1] https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

policy details, the dataset must be made smaller in order for the model to make more accurate predictions. Recursive Feature Elimination (RFE) is utilized to reduce the number of features used to train our model. This is to prevent features that have a random correlation from altering the data, which affects its accuracy. All features are initially included in training the model and RFE calculates the importance of each feature. The least important features are removed and the model is retrained with the remaining features. This process repeats until we find the most important variables to train our final model on.

Only the top six features are kept as the seventh most important feature Policy Index was found to have little to no effect on the model. It was computed that the sixth most important variable, State had an overall importance to our model of 0.075621, whereas Policy Index was at 0.032945. This significant difference is usage in our model resulted in us deciding to drop Policy Index and thus, we are left with our six features, Customer Lifetime Value, Income, Marital Status Index, Number of Policies, Days Since Effective Date, and State.

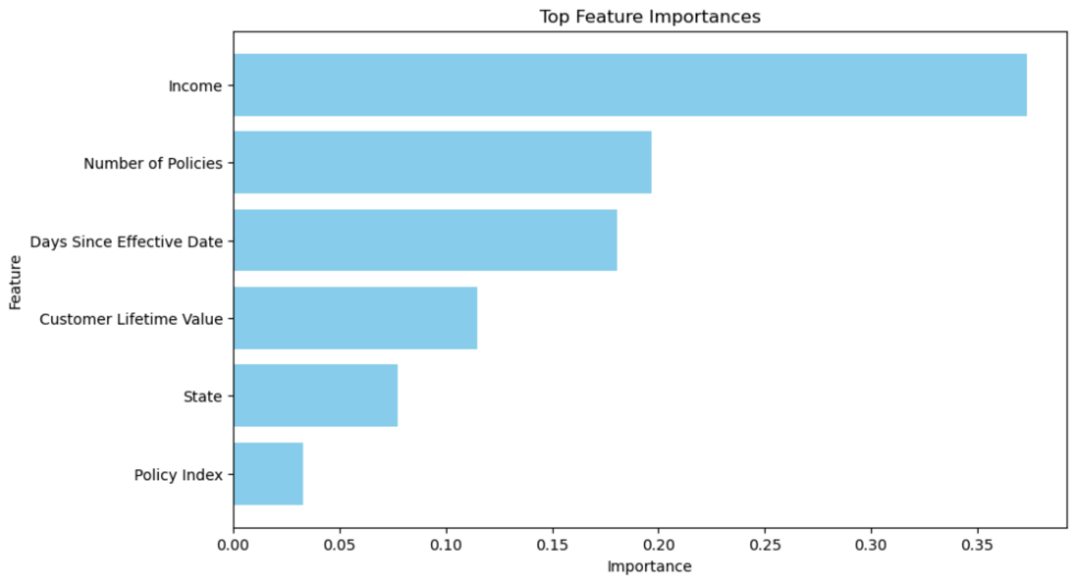

Figure 2: RFE Visualizer for selected features

3. Optimizing our Model[3]

Hyperparameter optimization is the process of finding the best set of settings to guide how the machine learns. In this case, we focused on adjusting certain settings, including how many models we combine, how deep each model can grow, the minimum number of samples needed to split a decision, and the minimum number of samples required to

[3] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

end a branch. We used a method called Grid Search, which tries every possible combination of these settings to identify the combination that produces the most accurate model. Once found, we use these settings to build our final model.

4. **The Model**
   After optimizing our model to the optimal parameters, we were able to complete it, finishing with an overall accuracy on the training data of 92.798%. This means that for every claim given predict whe in the dataset, our model was able to correctly predict whether a claim would be over $1000 or not about 93% of the time, which we felt was sufficient enough to use to predict on the test data. This new dataset did not include whether a claim was over $1000, but with this model, was able to correctly predict claims over $1000, 93.167% of the time.

Results and Discussion

The model has a high accuracy of 93%. Moreover, the use of multiple trees reduces overfitting compared to individual decision trees which makes our model reliable for predicting policies with unknown claims. Random Forest also provides a measure of feature importance, which helps in understanding which variables contribute most to the model's predictions, making it more transparent and explainable. This would be especially helpful for explaining the model to regulators who value model transparency and improve the rate at which company filings are approved.

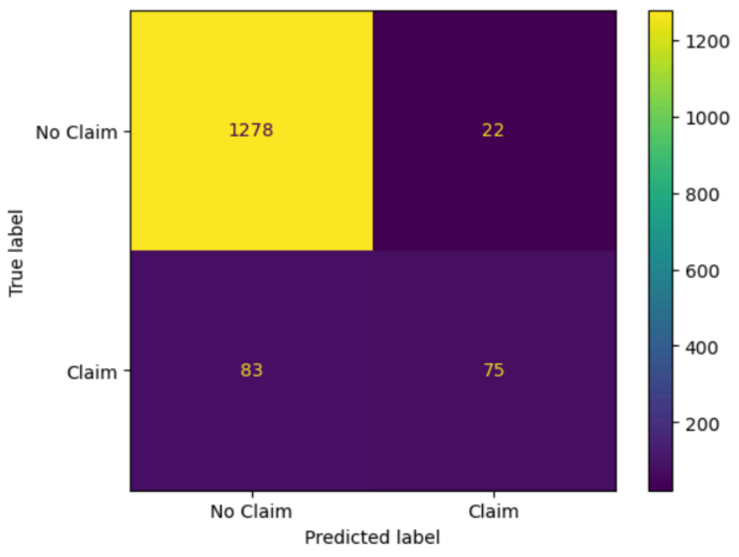

Figure 3: Confusion Matrix Visualizer

The model also has a very high true negative rate of 98.31% however, it has a subpar true positive rate of 47.47%. This indicates that while our model can predict claims less than $1000, the results for claims over $1000 are inaccurate about half the time. This is unsatisfactory since the detection of high-cost claims is more important for various insurance business operations to be discussed in the section below. This result can be explained due to a class imbalance in the dataset where only 832 out of the 6458 observations had claims over $1000. Therefore, the model is biased towards predicting no claim, as that is what was seen the most during training. To further improve the model. We can make class balancing adjustments to the data, such as random oversampling/undersampling or adjusting class weights. Furthermore, such models could also be trained on a more specific claims related dataset if available that contains more features such as Claim time, Claim status, and Claim duration, to reduce class imbalance and have a more evenly distributed training set.

Potential Applications

The insights from this model can be utilized in several ways.

1. **Reserving**
   By predicting claims over a certain threshold, the company can more accurately estimate potential payouts, allowing them to allocate reserves more effectively.

2. **Pricing**
   Being able to predict whether or not a policy will have a high cost claim based on different risk characteristics and policy information would improve pricing segmentation, allowing the company to differentiate between more and less risky customers to maintain pricing fairness and rate adequacy.

3. **Fraud Detection**
   Detecting high-cost claims allows it to be scrutinized more closely for potential fraud and thus help direct attention to claims that might warrant further investigation.

Conclusion

Through the use of Random Forest, Recursive Feature Elimination (RFE), and Grid Search, the model was able to identify the key factors that influence claims outcomes and achieved a high accuracy score of over 92% making it a reliable tool for actuaries and stakeholders in understanding and predicting claims, ultimately supporting their strategic decision-making in various actuarial applications.

This model provides business partners with a rigorous tool for evaluating claims risk, improving actuarial accuracy, and guiding automotive claims decision-making. The predictive insights generated by the model can enhance claim assessments and refine financial forecasting, creating a more effective approach to risk management. By incorporating this model into the claims process, we aim to deliver more reliable predictions for claims exceeding $1000. This model will prove to be useful for business partners predicting automotive insurance claims.

Code Appendix

Determine the best variables

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel

data = pd.read_csv('train.csv')
df_clean = data.drop(columns=['CustomerID'])

df_encoded = pd.get_dummies(df_clean, drop_first=True)

X = df_encoded.drop(columns=['Claim over 1k'])
y = df_encoded['Claim over 1k']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

sfm = SelectFromModel(rf, threshold='mean')
sfm.fit(X_train, y_train)

selected_features = X_train.columns[sfm.get_support()]

from sklearn.feature_selection import RFE

rfe = RFE(estimator=rf, n_features_to_select=10)
rfe.fit(X_train, y_train)

rfe_selected_features = X_train.columns[rfe.get_support()]

rfe_selected_features
```

Train the data

```python
rf = RandomForestClassifier(n_estimators=300, max_depth=10, min_samples_split=5,
                            min_samples_leaf=2, bootstrap=False, random_state=42)

rf.fit(X_train, y_train)

importances = rf.feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': X_final.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print("Feature Importances:\n", feature_importance_df)

top_features = feature_importance_df['Feature'][:6].values
X_train_top = X_train[top_features]
X_test_top = X_test[top_features]

rf_top = RandomForestClassifier(n_estimators=300, max_depth=10, min_samples_split=5,
                                min_samples_leaf=2, bootstrap=False, random_state=42)

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(estimator=rf_top, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_

best_rf.fit(X_train_top, y_train)

y_pred = best_rf.predict(X_test_top)

accuracy = accuracy_score(y_test, y_pred)
```

Optimize the Data

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import GridSearchCV

data = pd.read_csv('train.csv')

data['Effective to Date'] = pd.to_datetime(data['Effective To Date'], format='%m/%d/%Y', errors='coerce')

reference_date = pd.Timestamp('2023-01-01')
data['Days Since Effective Date'] = (data['Effective to Date'] - reference_date).dt.days

data['Days Since Effective Date'].fillna(data['Days Since Effective Date'].median(), inplace=True)

features = ['Sales Channel Index','Marital Status Index',
            'Customer Lifetime Value','Income','Number of Policies','Days Since Effective Date','State','Policy Index']

X = data[features]
y = data['Claim over 1k']

categorical_cols = ['State']

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_cols)
    ],
    remainder='passthrough'

X_preprocessed = preprocessor.fit_transform(X)

encoded_state_columns = preprocessor.named_transformers_['cat'].get_feature_names_out(['State'])

X_preprocessed_df = pd.DataFrame(X_preprocessed, columns=list(encoded_state_columns) + features[1:], index=X.index)

states_to_drop = ['State_California', 'State_Nevada', 'State_Arizona', 'State_Washington', 'State_Oregon']
states_to_keep = [col for col in encoded_state_columns if col not in states_to_drop]

X_final = X_preprocessed_df[states_to_keep + features[1:]]

X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=42)
```

Perform predictions on test data

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

test_data = pd.read_csv('test.csv')
test_data['Effective to Date'] = pd.to_datetime(test_data['Effective To Date'], format='%m/%d/%Y', errors='coerce')
test_data['Days Since Effective Date'] = (test_data['Effective to Date'] - reference_date).dt.days
test_data['Days Since Effective Date'].fillna(test_data['Days Since Effective Date'].median(), inplace=True)

X_test_new = test_data[features]

X_test_preprocessed = preprocessor.transform(X_test_new)

X_test_preprocessed_df = pd.DataFrame(X_test_preprocessed, columns=list(encoded_state_columns) + features[1:],
                                      index=test_data.index)

X_test_final = X_test_preprocessed_df.reindex(columns=X_train.columns, fill_value=0)

predictions = rf.predict(X_test_final)

submission = pd.DataFrame({
    'CustomerID': test_data['CustomerID'],
    'Claim over 1k': predictions
})

submission.to_csv('submission_predictions.csv', index=False)
```