State Farm Work Sample Summary

The objective of this work sample is to build a model aiming to identify if the applicant had a future loss (binary outcome classification) with the provided claim, predictor, subsequent loss experience dataset.

1. Data Preprocessing

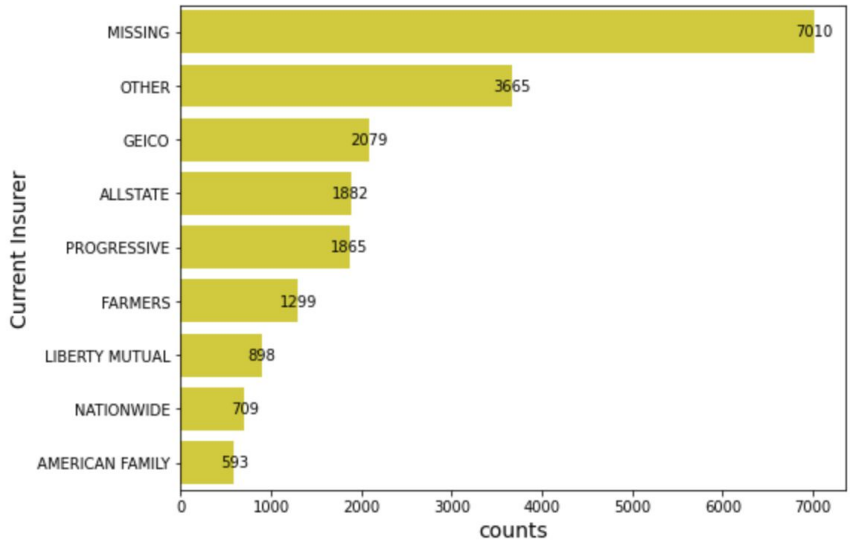
My initial step is to perform exploratory data analysis and data cleaning on the provided datasets:

(1) **Claim Data**: Given by the instruction, the criteria to count a claim is that it must have at least $1 paid out and be prior to the application date 1/1/2017. Hence, I would filter out claims before 1/1/2017 and amount paid for claims less than $1. Also, the aggregated table wants to focus on claims in each of the last 5 years from the application date (1/1/2017), so claims made before 1/1/2012 would be excluded.

(2) **Predicor Data**: it consists of 39 numerical features, 1 categorical feature, and 1 boolean feature, and I have done the following data wrangling:

(a) Convert boolean feature to numerical feature.

(b) Investigate the categorical feature 'curnt_insurer'. Though it doesn't contain Null, it contains a MISSING type. A MISSING type of current insurer corresponds to NA's in the 'time_w_carr' and 'prior_bi' column. It also correlates with the 'inforce_ind' feature where MISSING indicates a 0 in 'inforce_ind' and all other types of insurer indicates a 1. Thus, we can drop the 'inforce_ind' column to reduce multicollinearity.

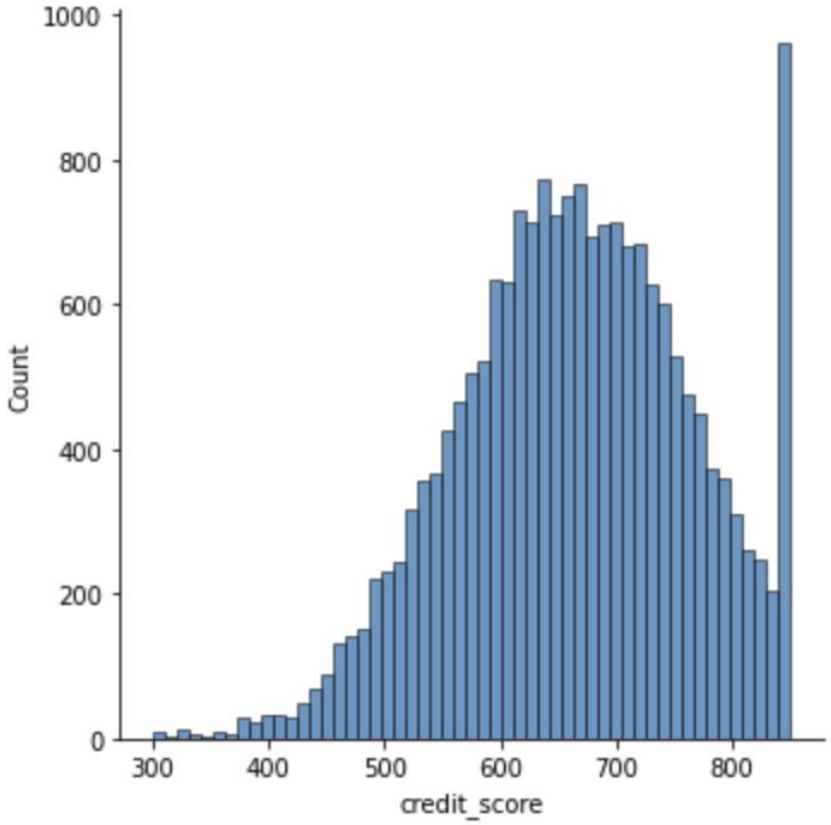


(c) Although NA's in 'time_w_carr' and 'prior_bi' features associate with the missing type in 'curnt_insurer', we want to assume that the missing values are missing at

random because the appearance of MISSING type has no explicit pattern in the data. As a result, we impute the missing values in 'time_w_carr' and 'prior_bi' with mean and median.

(d) Impute the missing values in the 'veh_lien_cnt' column using the formula: vehicle count = vehicle lien + vehicle lease + vehicle own, by assuming that a household can apply insurance on a vehicle if and only if the vehicle is liened, leased, or owned (fully paid) by the household.

(e) Impute NA's in the 'credit_score' column with mean since the credit scores appear to be normally distributed except for some at score 850, and the NA's can be assumed missing completely at random as it has no missing pattern.



(3) **Subsequence Loss Experience Data**: loss ratio is calculated by loss amount divided by premium, so we can drop either one of them to retain the same information. I choose to drop the loss amount (feature 'pd_amt').

2. Aggregated Table for Step 1.1

To create the aggregated model required by step 1.1, I first extract the year from the claim dates as shown below, and we only focus on years from 2012 to 2016

```
claims.head()
```

	hhld_id	clm_dates	aft_ind	pd_amt	Claim_Year
3	10347	2014-02-09	0	428.48	2014
6	2388	2016-08-01	1	2092.55	2016
8	8845	2015-08-10	1	6133.88	2015
11	14318	2015-11-26	0	157.36	2015
12	13283	2012-04-28	0	164.90	2012

The aggregated table is created by grouping on the household id and the year and the resulting aggregated table is shown below.

```
claims_agg.head()
```

	hhld_id	Claim_Year	at_fault_cnt	not_at_fault_cnt
0	2	2015	0	1
1	13	2016	1	0
2	27	2013	0	1
3	36	2013	0	1
4	49	2014	0	1

```
# sanity check
claims_agg[claims_agg.hhld_id == 8845]
```

	hhld_id	Claim_Year	at_fault_cnt	not_at_fault_cnt
1536	8845	2015	1	0

Assuming we would want to retain all our applicants information, we need to perform a left outer join using the **Predictor Data** as the main dataframe while merging **Subsequent Loss Experience Data**, **Predictor Data**, and the **aggregated data**. Since not all household id from **Predictor Data** appears in the **aggregated data**, there would be NA's for 'Claim_Year', 'at_fault_cnt' and 'not_at_fault_cnt' for household ids that do not appear in the **aggregated data**. To handle those NA's, I have done the following:

- Fill missing values with 0 in the 'at_fault_cnt' and 'not_at_fault_cnt' column because those households did not file any claim.

- By assuming that a household is equally likely to file claims at each year, we can drop the 'Claim_Year' feature because a claim at 2015 or a claim at 2016 makes no impact on whether a household would file a claim at the subsequent year after the application date.
- Also drop the household id because it has no information except serving as identification.

3. Training/Testing Data Preparation

Before proceeding to model building, we first need to prepare the training and testing data.

(1) Split the independent and dependent variables into X and y, and then use OneHotEncoding to convert the categorical feature 'curnt_insurer' into numeric because most of the machine learning models only work with numeric data types.

(2) Use train_test_split from Sklearn to split the data into 70% training set and 30% testing set. The reason not to have a validation set in this case is that the data is very unbalanced with 95% class 0 and only 5% class 1, and holding out a validation set would make test cases less and the class 1 cases even less in the test set.

(3) Use MinMaxScaler to normalize the training data, and use the same scale from training data to normalize the testing data. We perform normalization before SMOTE upsampling since SMOTE uses a distance-based algorithm to generate synthesized data, and normalizing the data would make SMOTE generate better results.

(4) Use SMOTE to upsample the minority class to create a balance training set. We do not upsample class 1 cases in the testing set because we want it to resemble the real life scenarios. The resulting training data has the following properties:

```
print(X_res.shape, y_res.shape)
```

```
(30988, 51) (30988,)
```

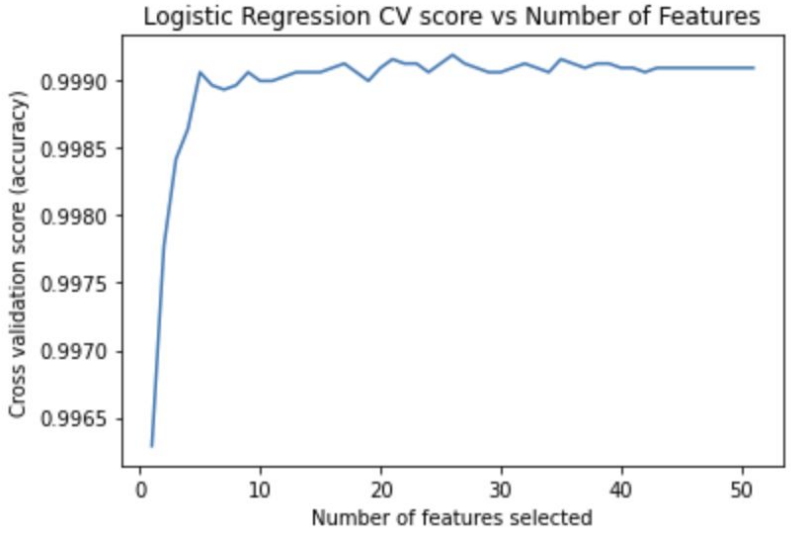
```
# Validate that we have a balance dataset
Counter(y_res)
```

```
Counter({0: 15494, 1: 15494})
```

4. Model Building

(1) Logistic Regression with Recursive Feature Reduction

I start with a base-line logistic regression model for this binary outcome classification task and I use RFECV to perform automatic recursive feature reduction at each iteration. According to the cross validation score shown below, having 7 features would start to give a high classification accuracy, and the highest accuracy is given when there are 26 features used in the logistic regression model.



The important features selected by RFECV are:

```
['drvr_cnt', 'veh_cnt', 'min_age', 'avg_age', 'min_mon_lic',
 'max_mon_lic', 'avg_mon_lic', 'cnt_yth', 'cnt_male', 'hoh_married',
 'cnt_auto', 'cnt_minr_viol', 'cnt_lic_susp', 'prior_bi', 'veh_lien_cnt',
 'veh_lease_cnt', 'veh_w_coll_cnt', 'veh_w_ers_cnt', 'credit_score',
 'premium', 'loss_ratio', 'at_fault_cnt', 'not_at_fault_cnt',
 'curnt_insurer_FARMERS', 'curnt_insurer_LIBERTY MUTUAL',
 'curnt_insurer_MISSING'],
```

And the metrics are:

```
Recall:  0.9943181818181818
AUC:  0.9955223048763318
Accuracy:  0.9992592592592593
```

(2) RandomForest

To find the best hyperparameters for the RandomForest model, I use RandomizedSearchCV from Sklearn. Some of the hyperparameters I used are n_estimators, max_depth and max_features. The best hyperparameters calculated from cross validation are:

```
{'max_depth': 100,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 233}
```

I used the Importance score from RandomForest to perform feature selection, and for illustration purpose, the top 20 most important features are:

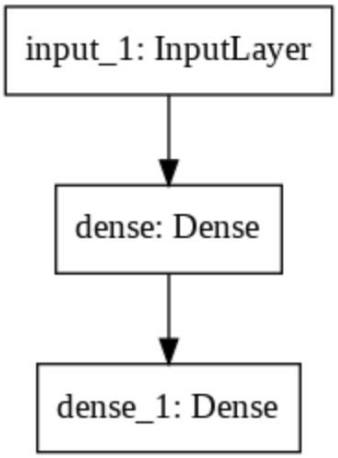
	importance
loss_ratio	0.412993
prior_bi	0.235658
credit_score	0.167369
currnt_insurer_MISSING	0.057692
avg_mon_lic	0.008561
veh_w_ers_cnt	0.008313
veh_w_comp_cnt	0.008256
max_age	0.008111
veh_w_coll_cnt	0.008053
veh_lien_cnt	0.007934
premium	0.006671
avg_age	0.006653
veh_own_cnt	0.006290
max_mon_lic	0.005659
hoh_age	0.005180
cnt_male	0.004465
cnt_female	0.004191
hoh_mon_lic	0.004186
time_w_carr	0.003429
currnt_bi_upp	0.003427

The RandomForest model gives a classification report as followed:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3874
1	1.00	1.00	1.00	176
accuracy			1.00	4050
macro avg	1.00	1.00	1.00	4050
weighted avg	1.00	1.00	1.00	4050

(3) Neural Network

For the Neural Network, I used a dense layer with 'ReLu' activation function and an output layer with 'Sigmoid' activation function (model shown below). I also use EarlyStopping to prevent overfitting by keeping track of the validation accuracy in each epoch.



The NN model gives a classification report as followed:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3874
1	1.00	0.99	1.00	176
accuracy			1.00	4050
macro avg	1.00	1.00	1.00	4050
weighted avg	1.00	1.00	1.00	4050

5. Conclusion and Future Improvement

All three models provide very similar accuracy and recall. Given this situation, I choose to use RandomForest as my final model because RandomForest allows an extensive amount of hyperparameters tuning while logistic regression does not, and RandomForest has a better interpretability compared to Neural Network. According to the feature importance table from RandomForest, ‘loss_ratio’, ‘prior_bi’, and ‘credit_score’ appear to be the most correlated features with whether the applicant had a future loss. As a result, we need to further investigate those features of an applicant while deciding on their premium. To get a more generalizable model, we should collect more data on people who had a future loss.