Loan Default Prediction

Benchmarking Classification Algorithms and Feature Selection Methods with Anonymized data

Data

Imperial College of London Loan Data

- 105471 samples
- 767 anonymized features

Business Context

- Assisting financial institutions
- Identifying users who are at risk of defaulting on loan
- Decrease loss of revenue from loan defaults
- Institute methods of intervention to reduce losses

Project Lifecycle

- Clean data
- Create data sets with feature selection methods
  - Filter
  - Random forest feature selection
  - Dimension reduction with PCA
- Exploratory data analysis
- Statistical analysis
- Predictive modeling

Data Cleaning Steps

Removing Problematic Columns

- Categorical variables with high cardinality
- Feature interactions with odd behavior
- Mixed data type columns

Feature Selection Methods

Filter Methods to create new data set

- Correlate features to target variable and remove below threshold
- Identify constant/quasi constant features for removal
- Reduction of column space from 767 to 52 features

Random Forest feature selection

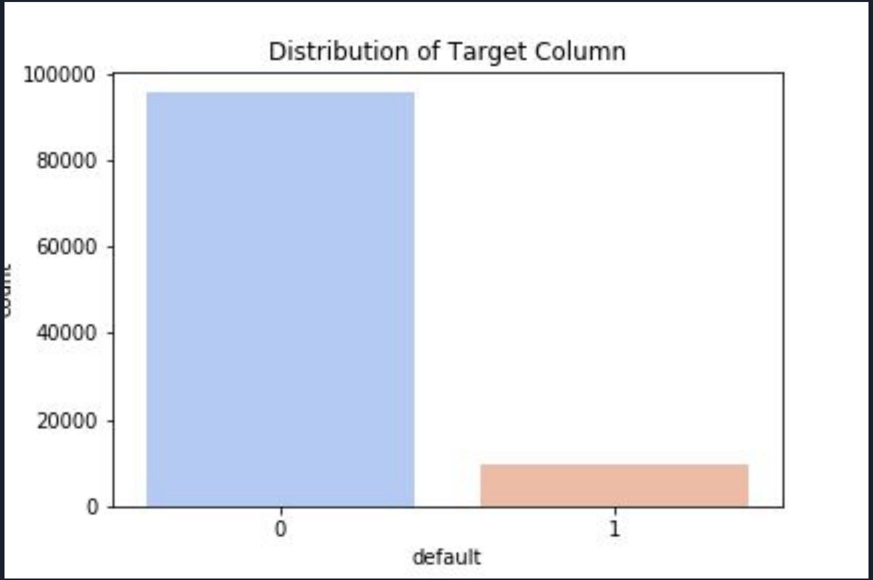
- Utilize ‘select_from_model’ to capture features thought to be important
- Reduction of column space to 333 features

PCA dimension reduction

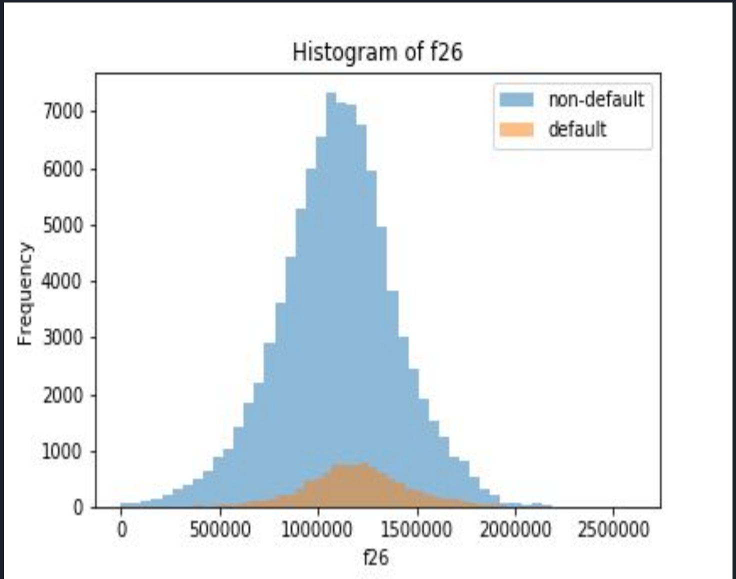
- Apply principal component analysis to reduce dimensions keeping n= 175 principal components

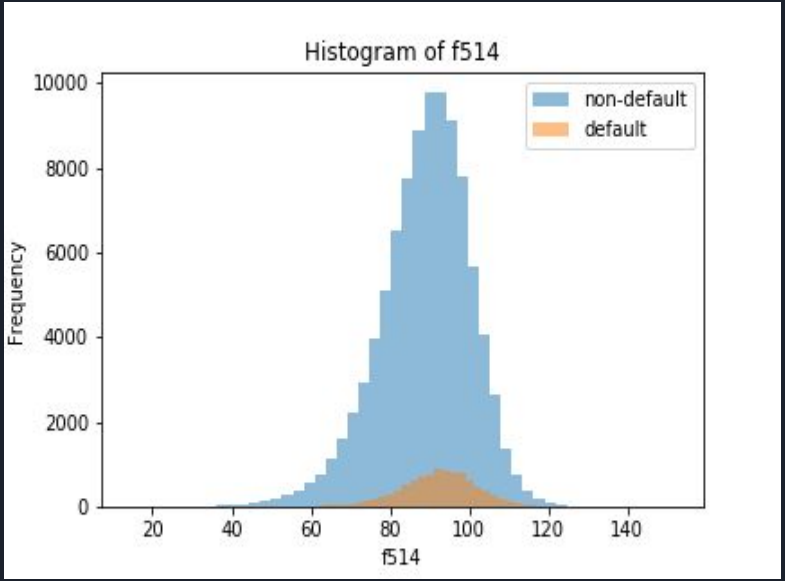
Exploratory Data Analysis

Target variable

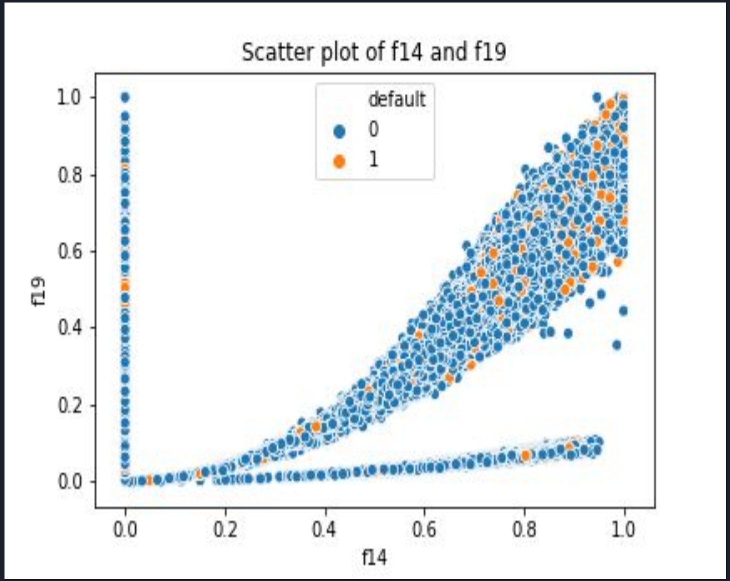


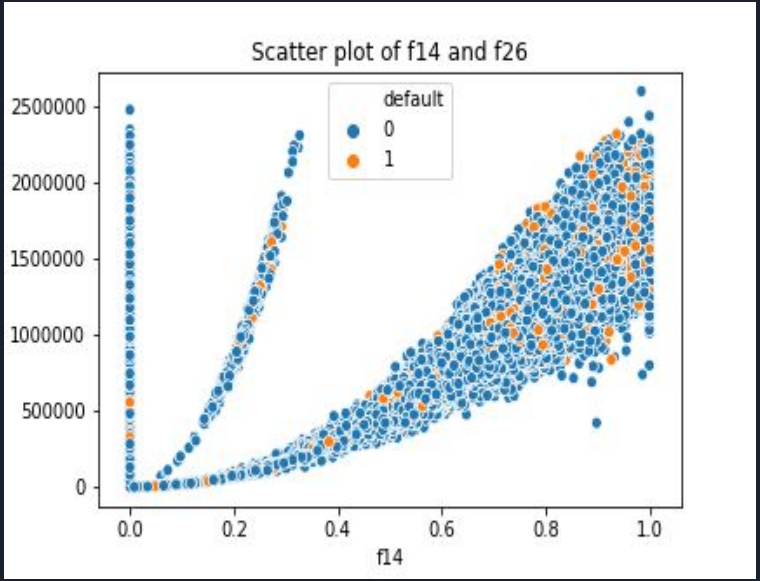
Comparing Selected Features Against Target



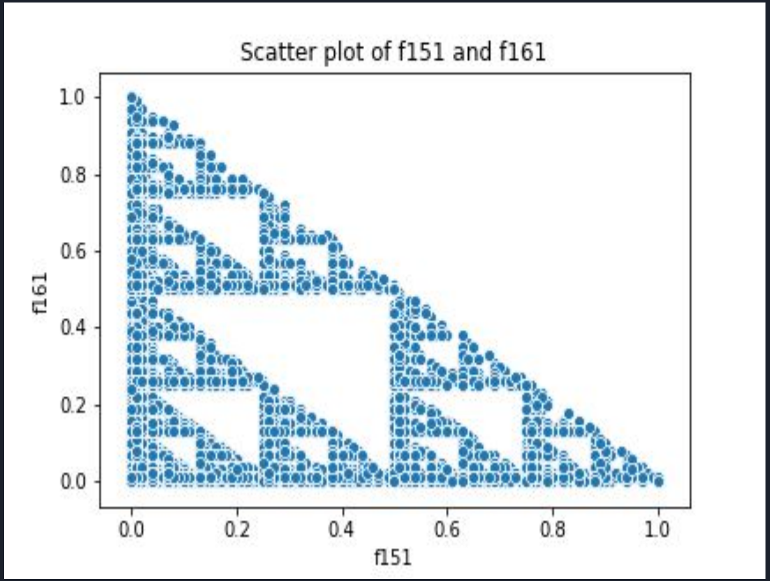


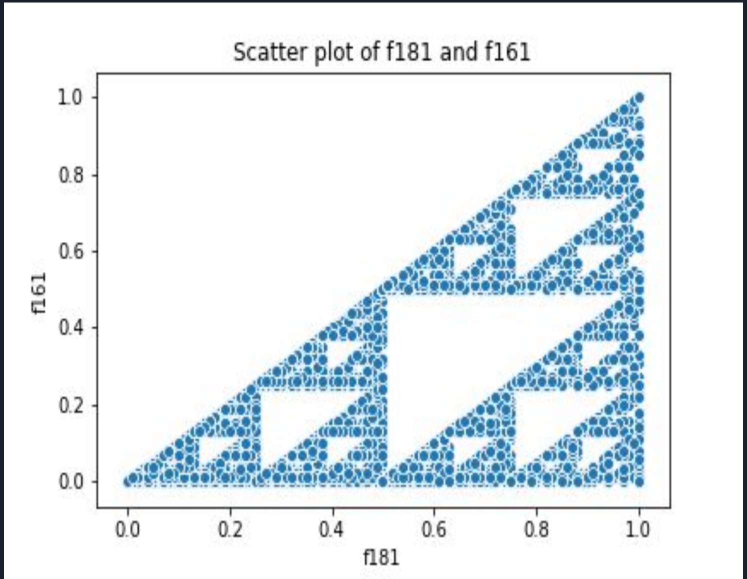
Comparing Selected Feature Interactions With Target



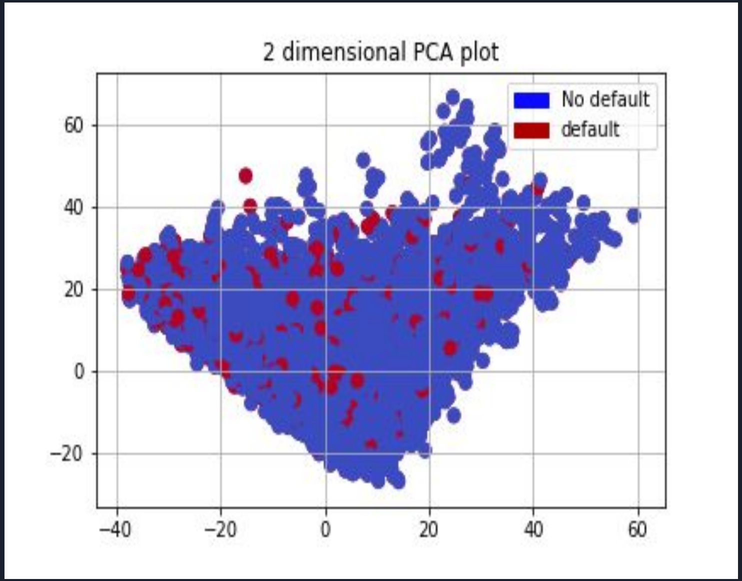


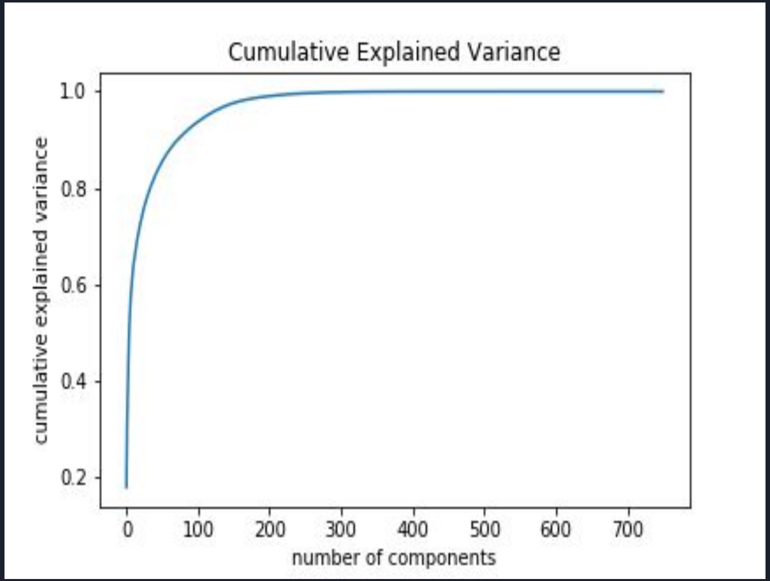
Identifying Unnatural Feature Interactions





PCA Features with Target





Statistical Analysis

Two- way t-tests on selected variables for default/non-default populations

- 3 variables tested for deviation of mean between defaulted/non-defaulted loanees
- In all cases P values extremely small
- Reject null hypothesis with strong confidence point estimates are unequal

Principal Component Analysis

- Determine number of components that capture 90% of variance

Predictive Modeling

Algorithms

- Logistic Regression
- ADAboost
- Random Forest
- XGBoost

Modeling Steps

- Create pipeline to run models for each of three datasets
- Set up search parameter grid
- Create random search object and fit pipeline
- Cross validation inside random search object
- Model evaluation
- Select best model/data set for further hyper-parameter tuning

Parameter Grid

```
# Create parameter grid for models and hyperparameters
grid_param = [
                {"classifier": [LogisticRegression()],
                 "classifier__penalty": ['l2','l1'],
                 "classifier__C": [100, 10, 1.0, 0.1, 0.01]
                 },
                {"classifier": [AdaBoostClassifier()],
                 "classifier__n_estimators": [50,100,150,200],
                 "classifier__learning_rate": np.arange(0,1,.01)
                 },
                {"classifier": [RandomForestClassifier()],
                 "classifier__n_estimators": [10, 100, 1000],
                 "classifier__max_depth":[5,10,15,25,30,None],
                 "classifier__min_samples_leaf":[1,2,5,10,15,100],
                 "classifier__max_leaf_nodes": [2, 5,10]}]
```

Model Results: Filtered Features

Hyper-parameters

ne	n estimators
lr	leanring rate
md	max depth
csbt	col sample by tree
γ	γ
spw	scale pos weight
msl	min samples leaf
mln	min leaf nodes

model	ne	lr	md	csbt	γ	spw	msl	mln	**auc train**	**auc test**
xgboost	150	0.1	4	0.2	0.1	9			0.73	0.67
adaboost	200	.56							0.697	0.668
random forest	10		none				2	10	n/a	0.637

Model Results: Random Forest Features

model	ne	lr	md	csbt	γ	spw	msl	mln	**auc train**	**auc test**
xgboost	150	0.1	4	0.2	0.1	9			0.801	0.706
adaboost	200	.34							0.732	0.697
random forest	10		30				5	10	n/a	0.670

Model Results: PCA Features

model	ne	lr	md	csbt	γ	spw	msl	mln	penalty	C	**auc train**	**auc test**
logistic regression									L2	10	0.718	0.699
adaboost	200	.24									n/a	0.690
xgboost	150	0.1	4	0.2	0.1	9					0.73	0.676
random forest	1000		none				2	10			n/a	0.637

Improving Model With SMOTE

- Select best model and data set for smore application: Xgboost, random forest features
- Create SMOTE pipeline class to upsample within pipeline
- Evaluate SMOTE model

SMOTE Model Results

Xgboost with SMOTE- Random Forest Features

Train/Test AUC: 0.677, 0.635 - Slightly worse than other models in regards to AUC

Precision/Recall on positive class (default): 0.141, 0.720

Improvement in recall of 0.9 from best model without SMOTE

Recommendations

Xgboost algorithm with random forest selected features with up-sampling

- Minimizes false negatives

Use for identification for loanees at risk of default

- Identify at risk individuals and institute a method of intervention