Medicare Provider fraud detection

JUAN ZHU

WHY ?

01

US spends $4.3 trillion on healthcare in 2021, accounting for 18.3% of the Gross Domestic Product (GDP). Each person costs $12,914 per year.

02

Medicare spends $900.8 billion, accounting for 21% of total national health expenditure (NHE).

03

Healthcare Fraud is estimated by the US Federal Bureau of Investigation to be 3% to 10% of overall spending.

Types of Healthcare Provider Fraud

- Phantom Billing.
- Unnecessary Services.
- Upcoding.
- multiple-billing.
- Unbundling.
- False price reporting.

Research Statement

Focus on healthcare fraud committed by provider.

Explore data analysis by using Medicare claims dataset.

Identify healthcare fraud indicators and fraudulent provider characteristics.

Build Machine learning classification models to predict potential providers.

Medicare claims datasets

Beneficiary

- Beneficiary ID
- Date of Birth
- Date of Death
- Gender
- Race
- Chronic Diseases Risk
- Annual Reimbursed Amount
- Annual Deductible Amount
- State
- County

Inpatient

- Claim ID
- Beneficiary ID
- Provider ID
- Claim Start & End Date
- Admission Date
- Discharge Date
- Attending Physician
- Operating Physician
- Other Physician
- Claim Reimbursed Amount
- Claim Deductible Amount
- Diagnose Codes
- Procedure Codes
- Diagnose Group Code

Outpatient

- Claim ID
- Beneficiary ID
- Provider ID
- Claim Start & End Date
- Attending Physician
- Operating Physician
- Other Physician
- Claim Reimbursed Amount
- Claim Deductible Amount
- Diagnose Codes
- Procedure Codes
- Admit Diagnose Code

Provider

- Provider ID
- Whether fraud

Workflow

Data Pre-processing

Data cleaning;

Feature Engineering;

Feature Selection;

Imbalanced Data Resampling.

Exploratory Data Analysis

Class label

Beneficiary Basic Information study

Fraud vs. non-fraud provider study

Exploratory Base Algorithm

Pipeline : feature Selection +Classifiers;

Algorithm Comparison

Optimization Model

Hyperparameter Tuning;
Evaluate Precision, F1, Recall;

Maximize Recall;

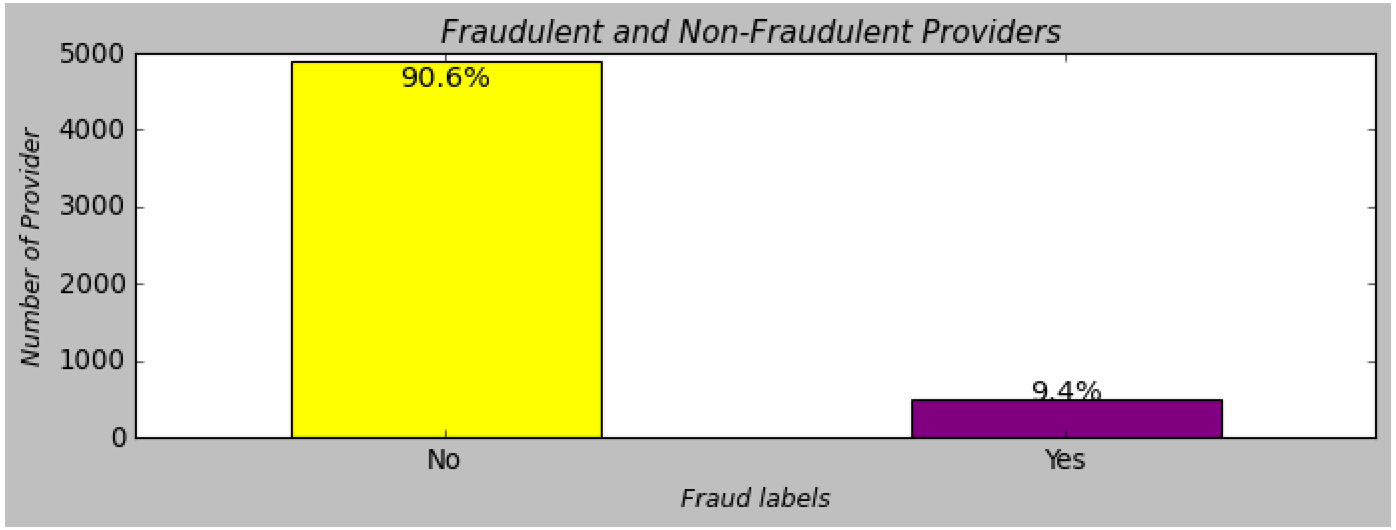
Feature importance Explain.

DATA PREPROCESSING

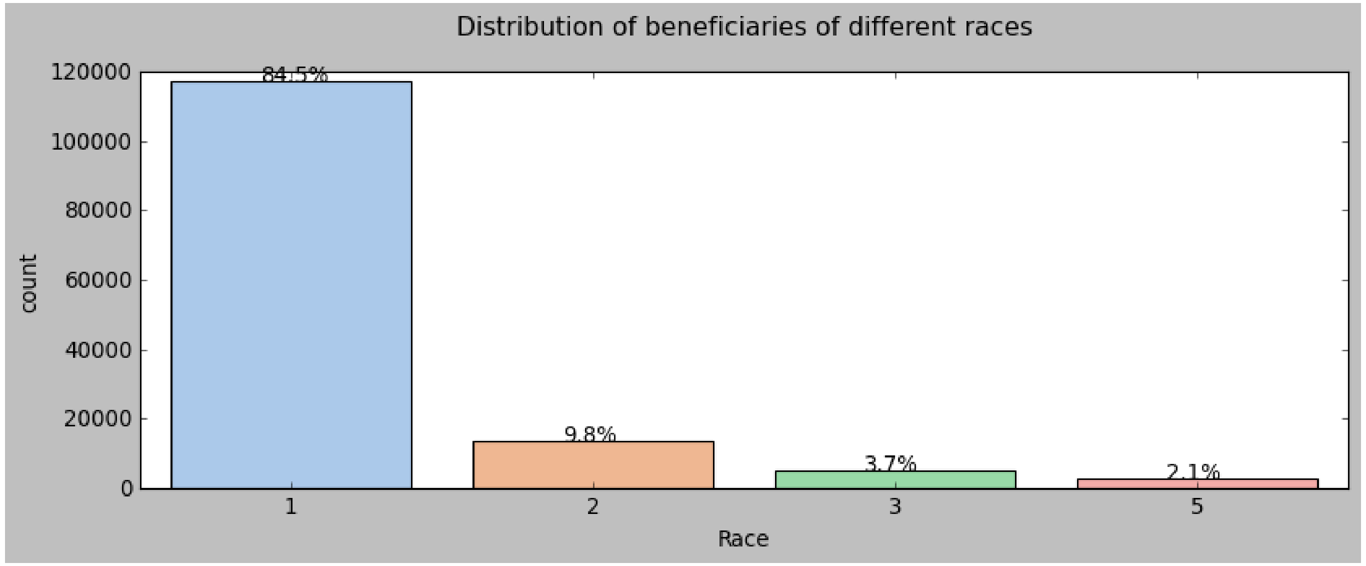
Imputing Missing Data :	Beneficiary: Inpatient: Diagnosis/ Procedure codes Outpatient: Operating physicians
Creating New Features :	Deceased, Tot_Reimbursed_Amt, Hospital_Stay, Claim_Duration, Physician_Count, Claim_Count, Chronic_Disease_Count, Diag_Code_count, Diff_Reimbursed_SC, etc.
Dropping features:	Features with high null values; Features from which new features were created, etc.
Integrating datasets	Integrate beneficiary, inpatient, outpatient and providers datasets by merging/concatenating on the common columns.
Label Encoding :	Categorical features
Data Scaling:	Robust-scaled: Due to the inclusion of outliers.
Handling Class Imbalance:	SMOTE, SMOTE +ENN.

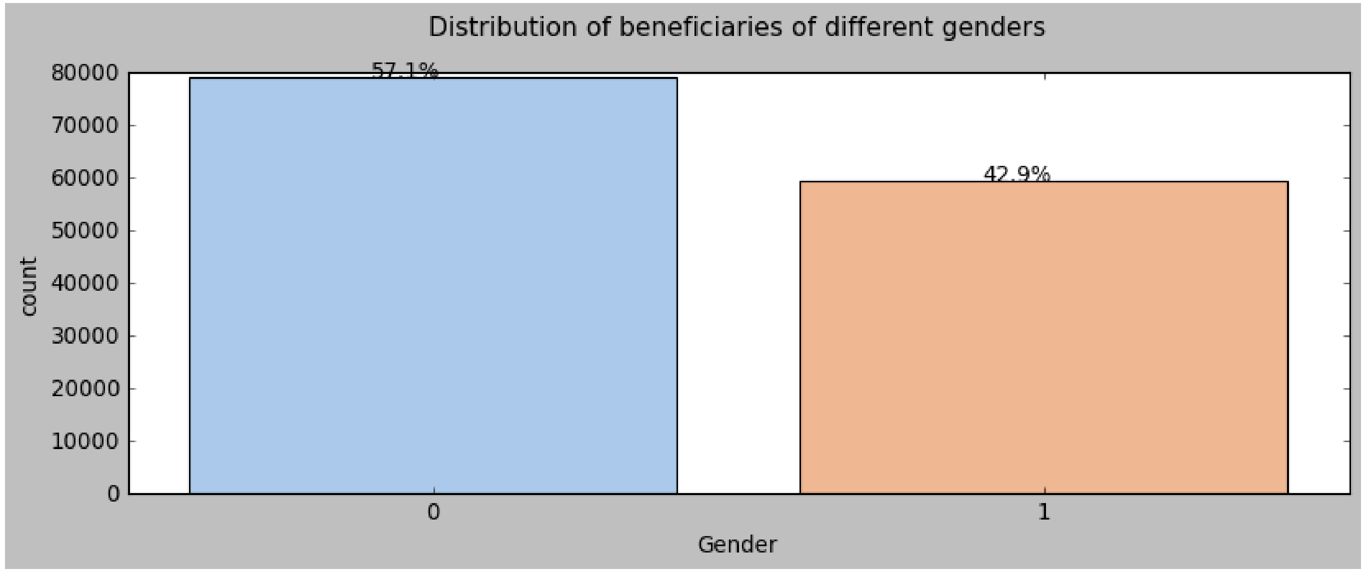
Exploratory Data Analysis

1. Class Label
Imbalanced data

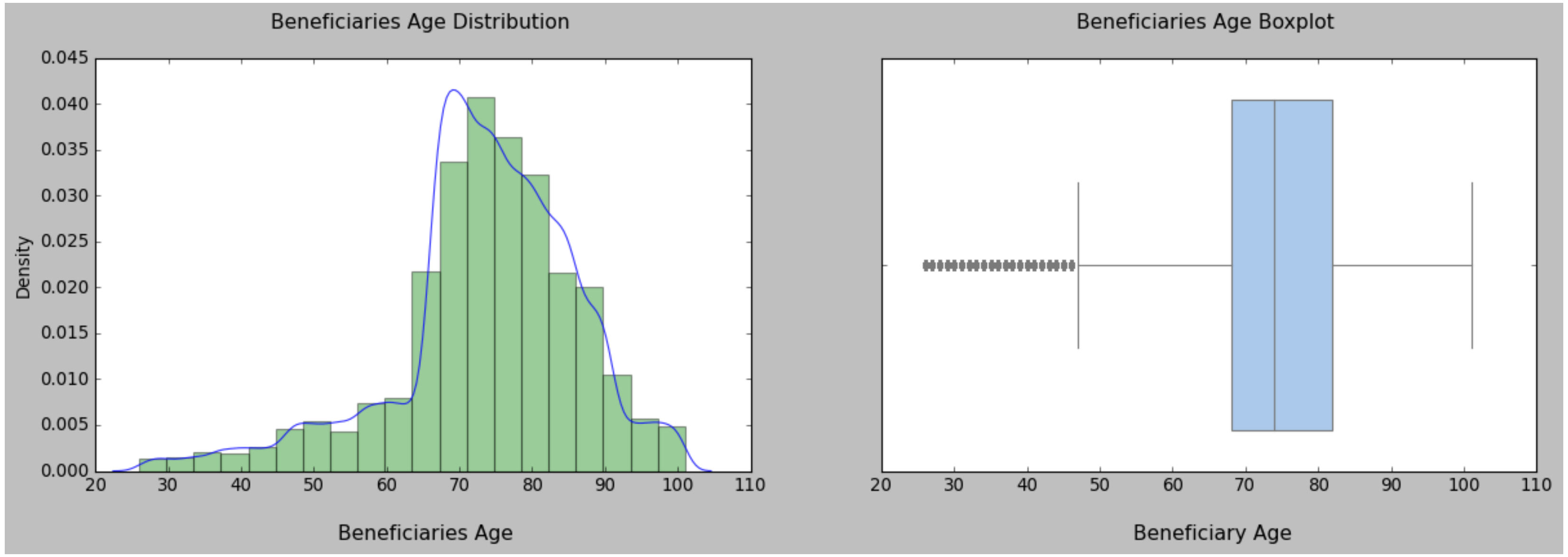


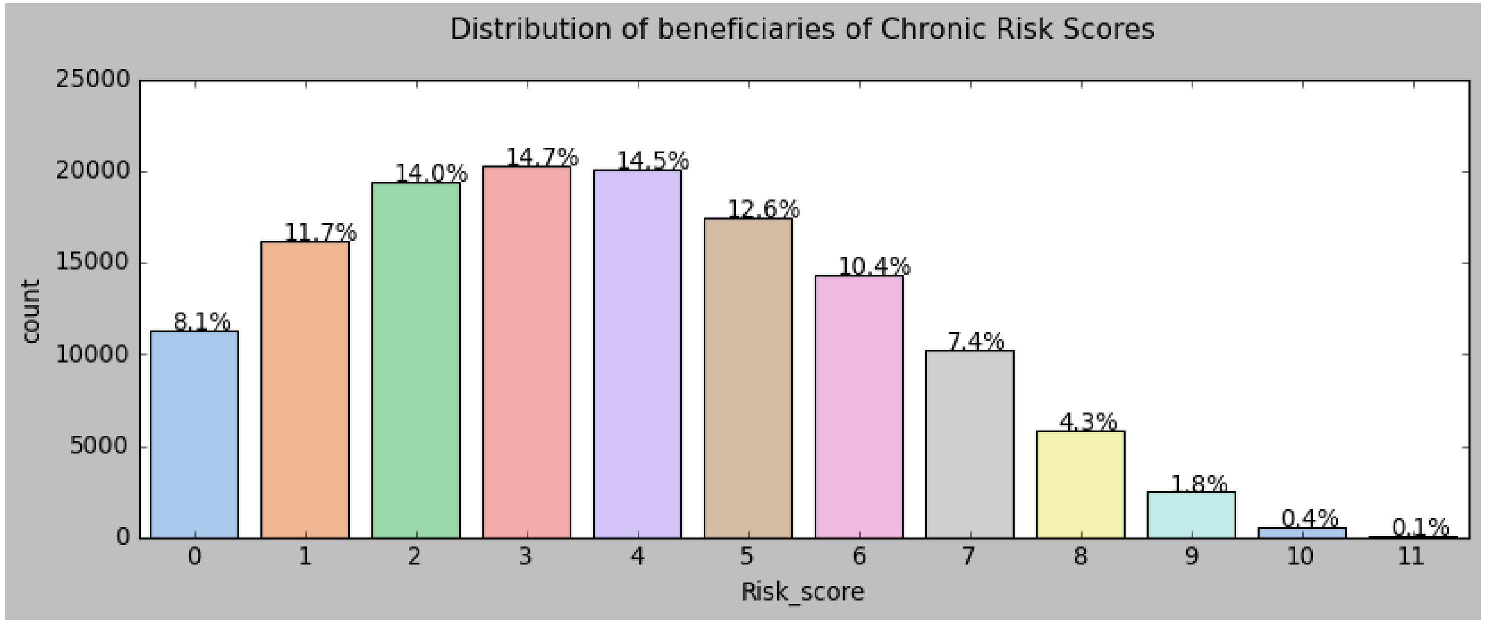
2. Beneficiary Basic Information Study



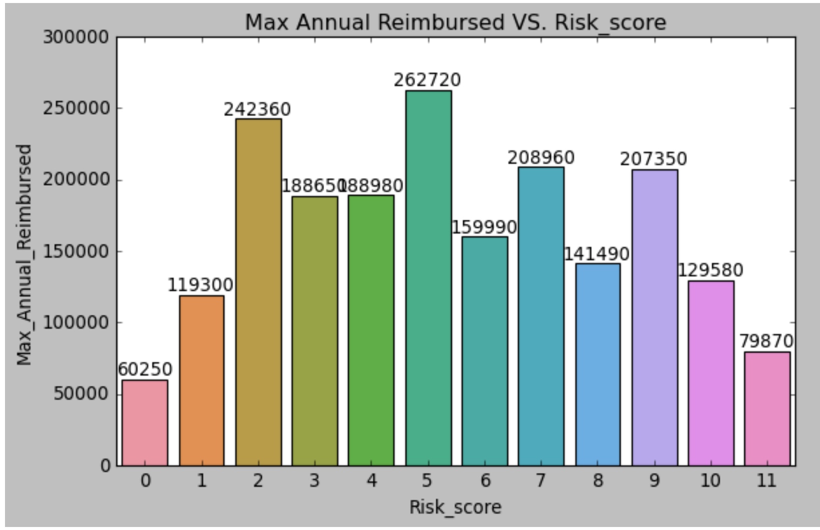


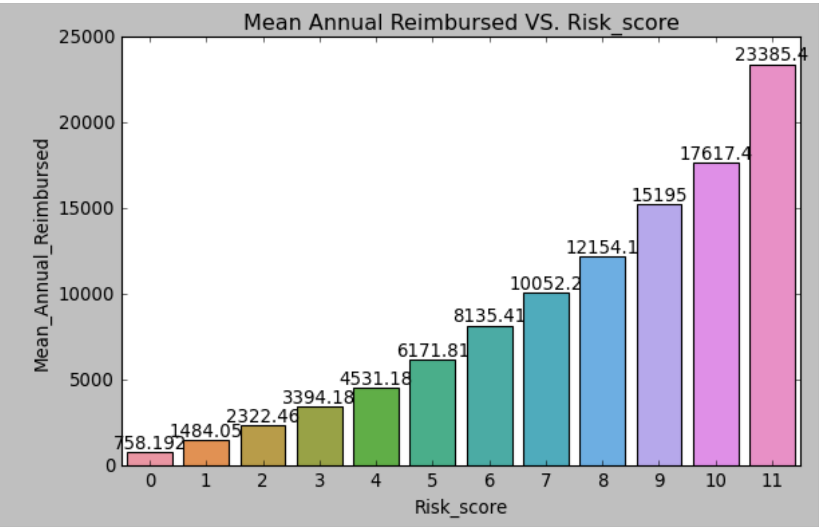
Beneficiary Basic Information (continue)

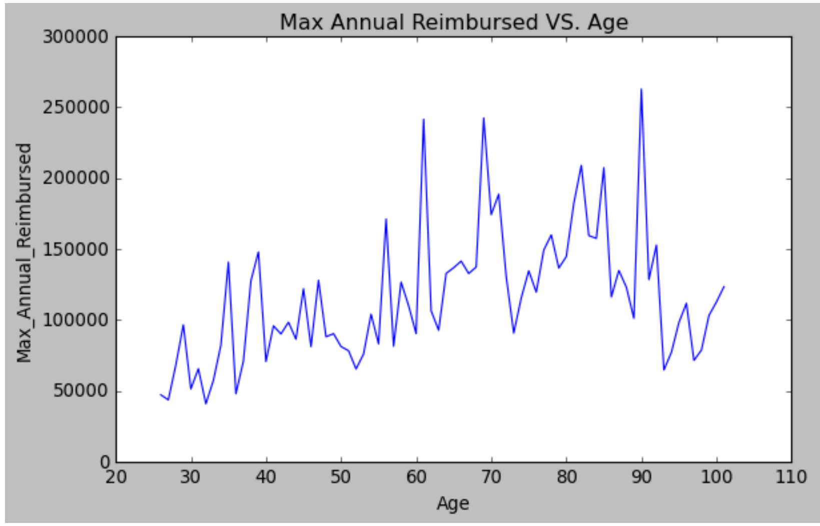


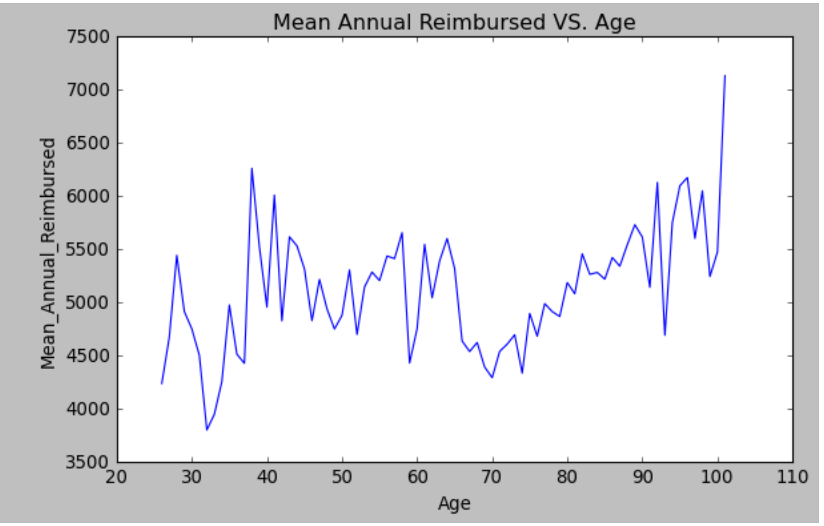


Beneficiary Basic Information (continue)









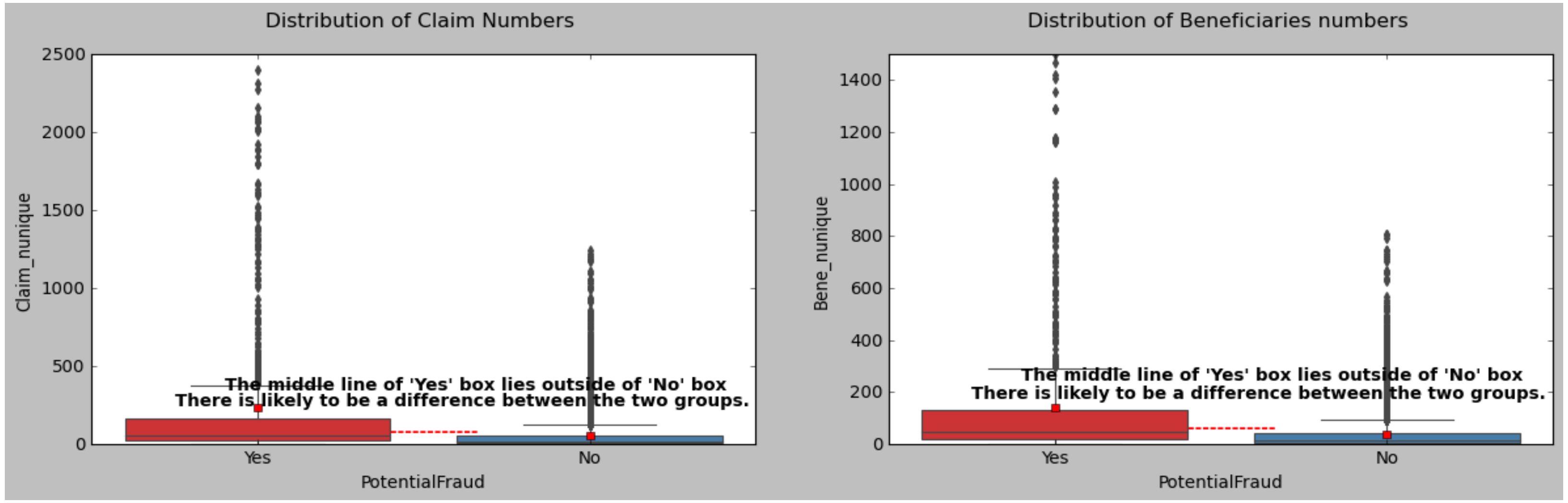
3. Fraud vs. non-fraud provider study

```
provider_group_integrated.Bene_nunique[provider_group_integrated['PotentialFraud']=='Yes'].median()
```

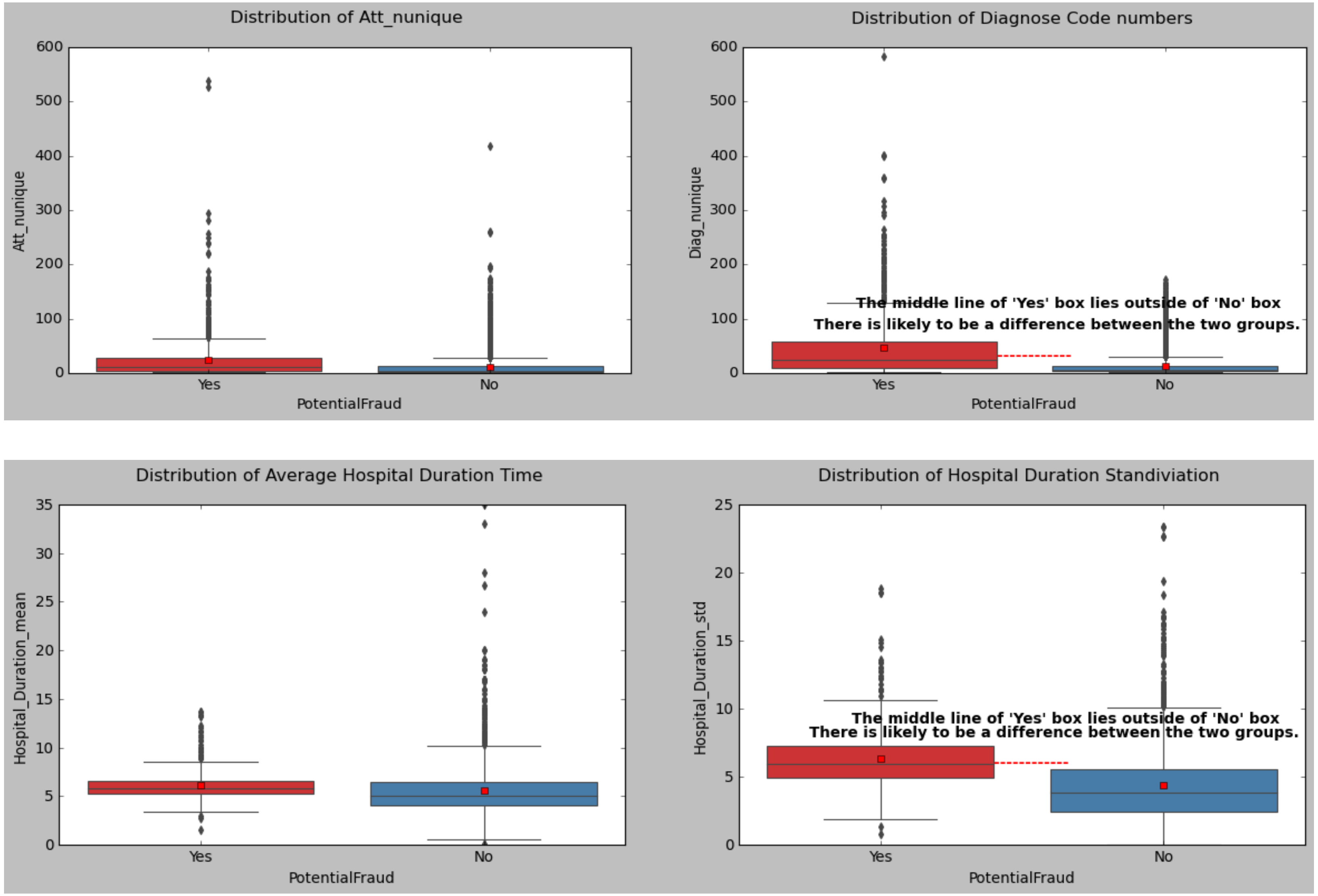
```
49.0
```

```
provider_group_integrated.Bene_nunique[provider_group_integrated['PotentialFraud']=='No'].describe()
```

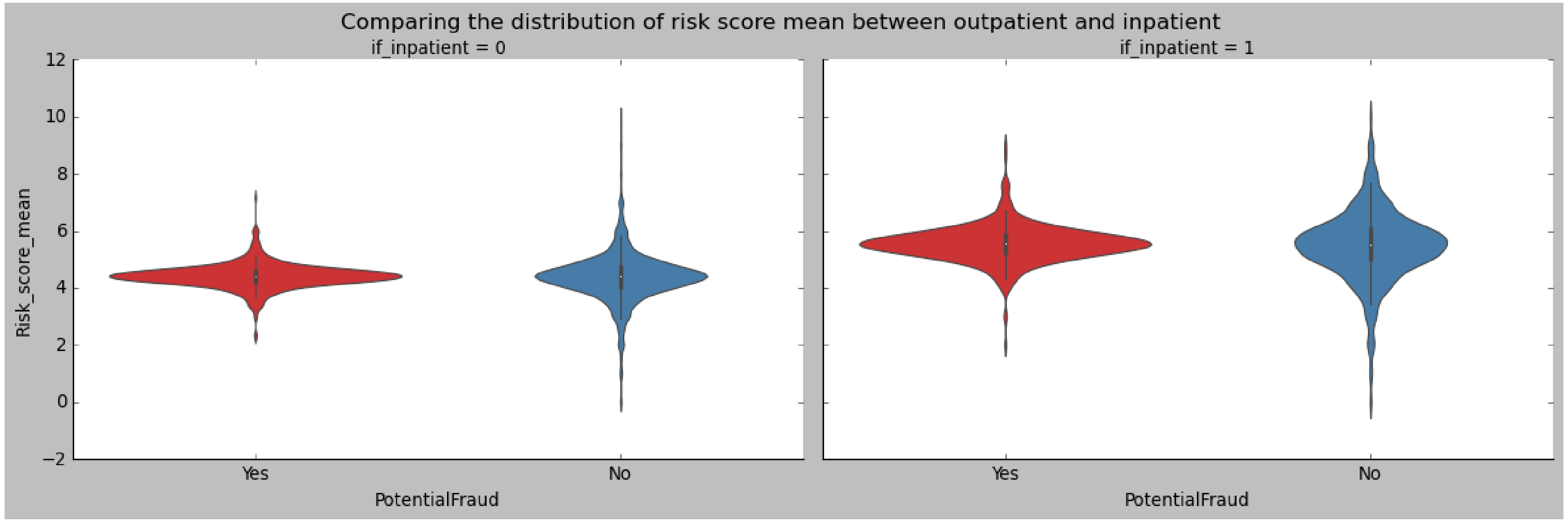
```
count    6202.000000
mean       39.049984
std        73.093235
min         1.000000
25%         5.000000
50%        15.000000
75%        40.000000
max       807.000000
Name: Bene_nunique, dtype: float64
```

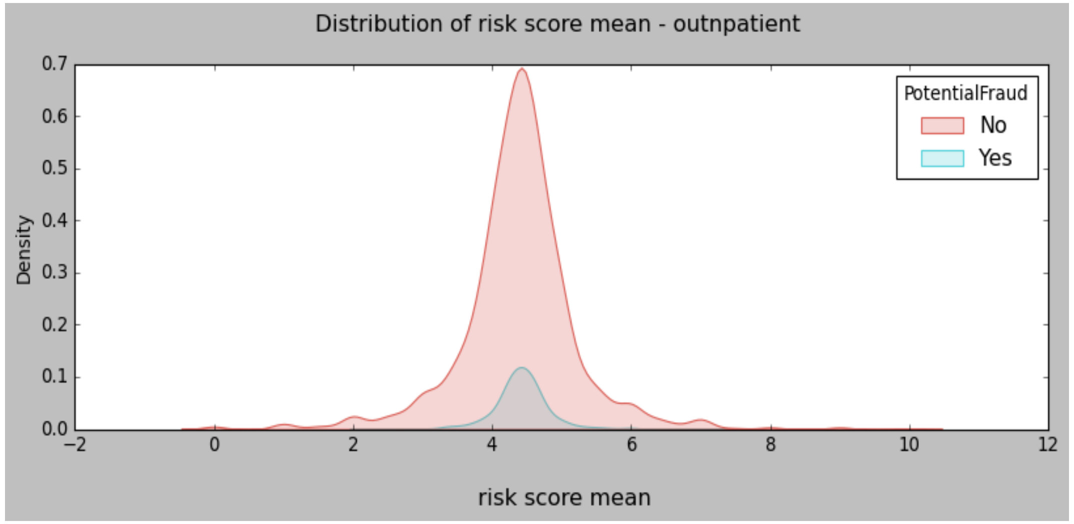


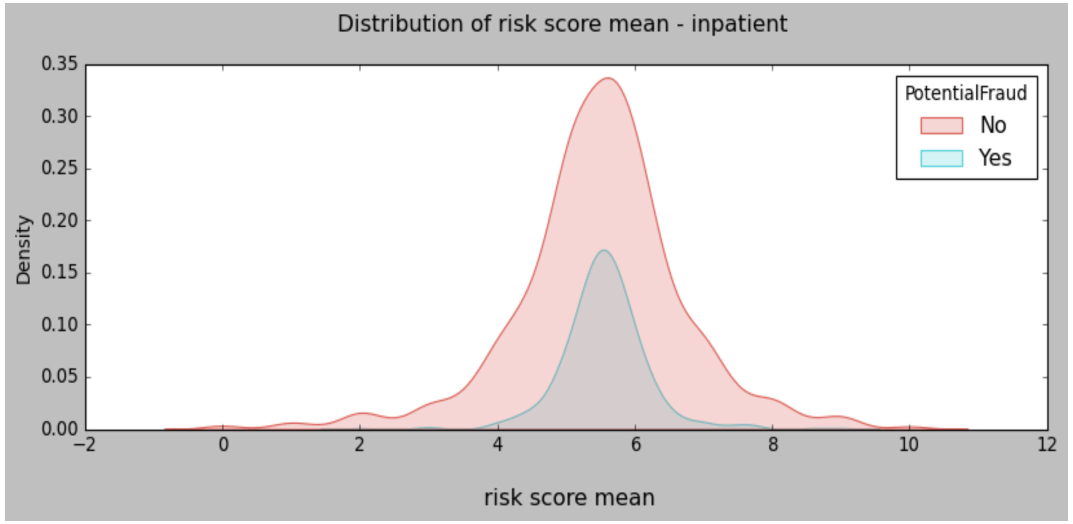
Fraud vs. non-fraud provider study (continue)



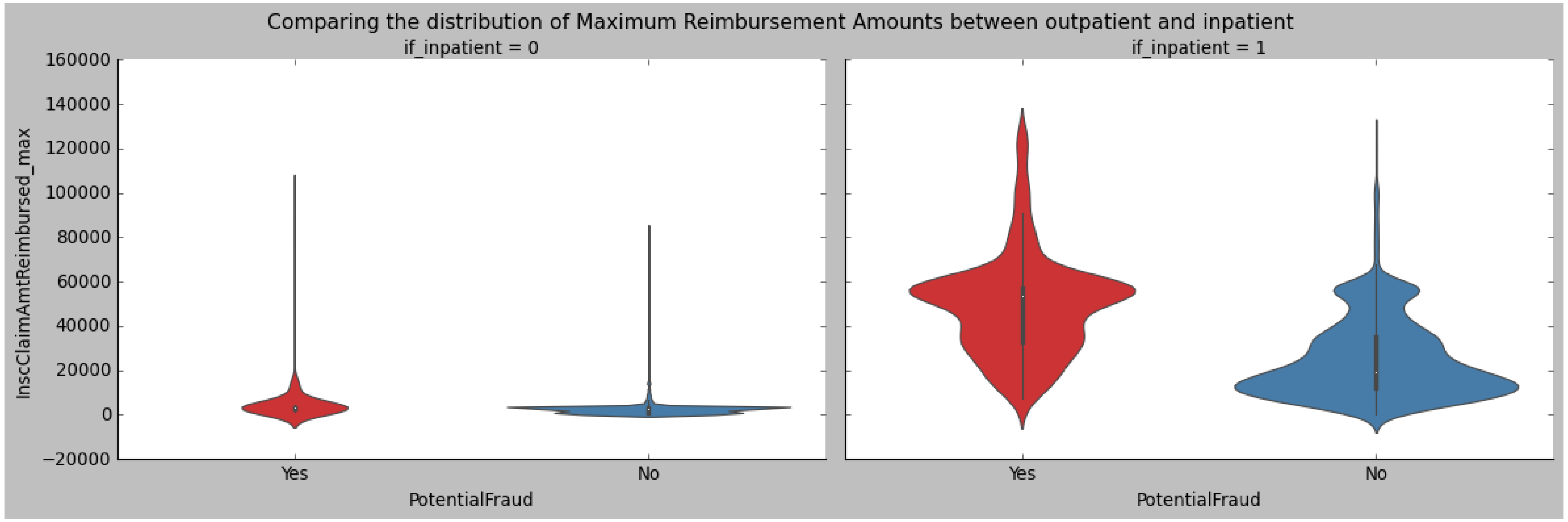
Fraud vs non-fraud provider study (continue)

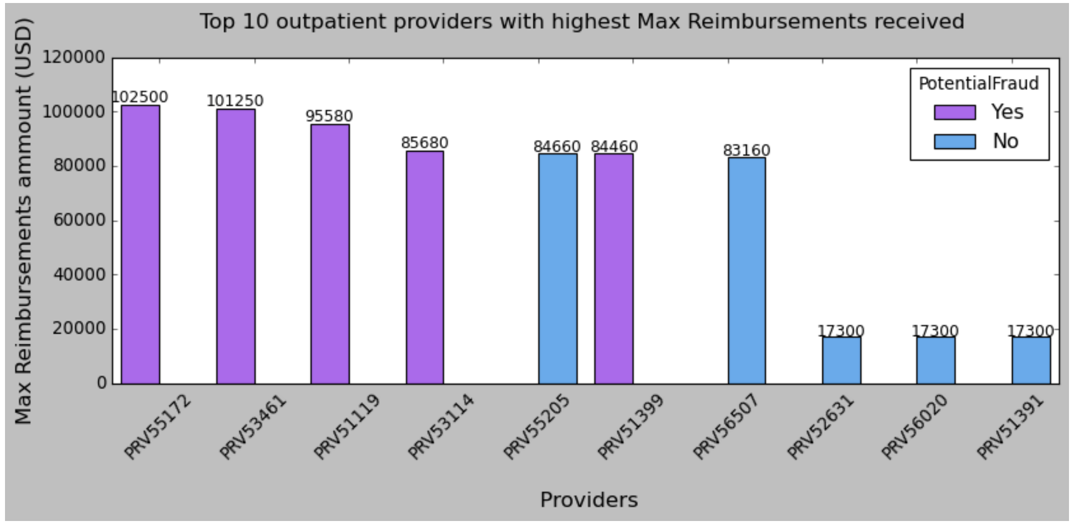


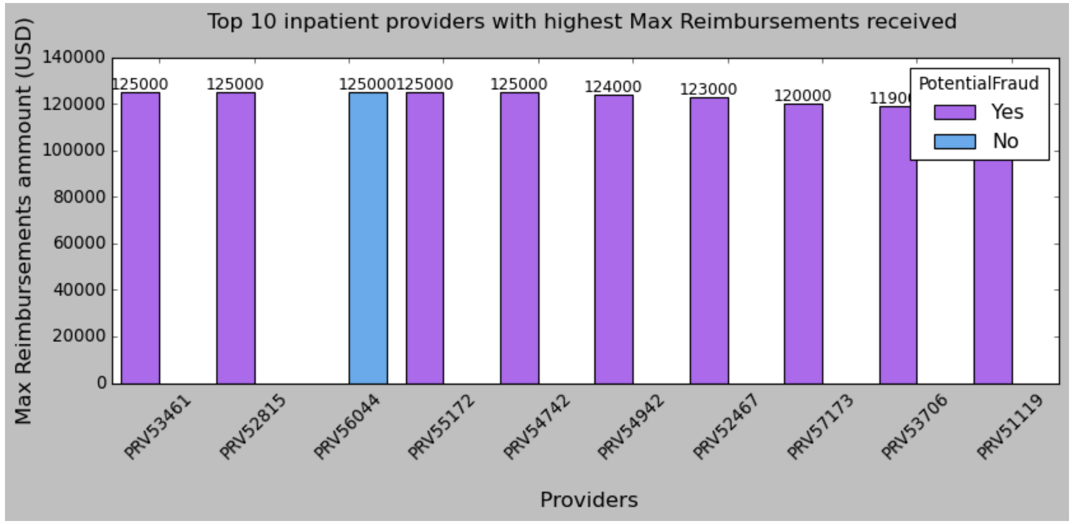




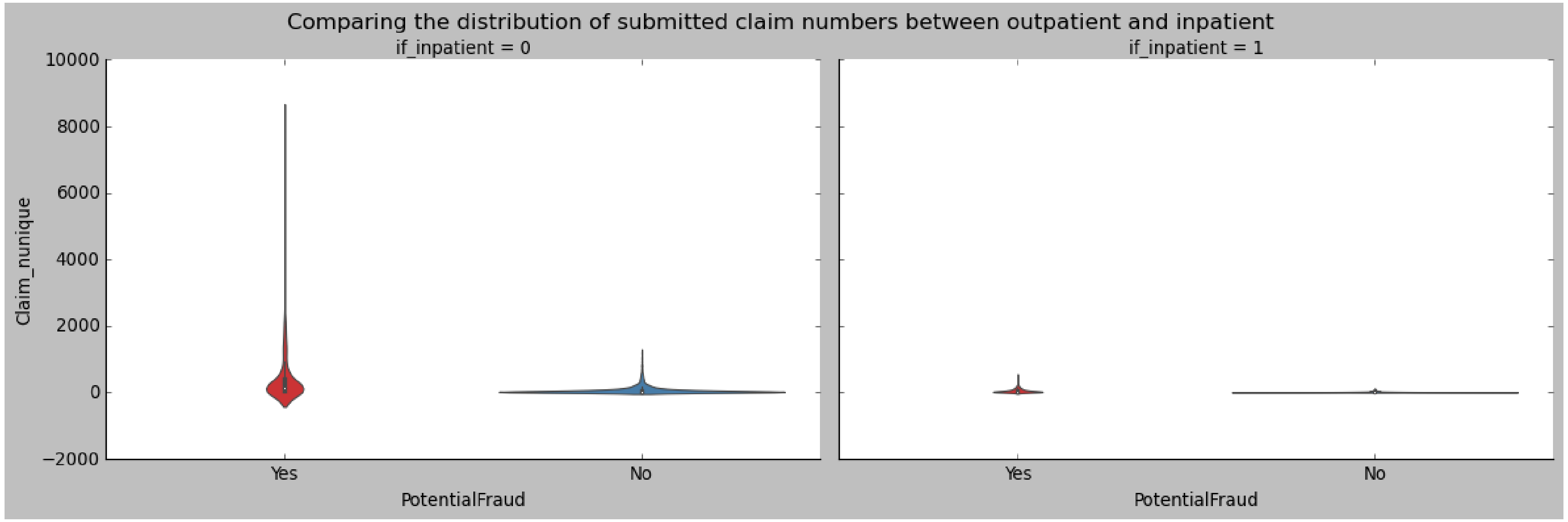
Fraud vs non-fraud provider study (continue)

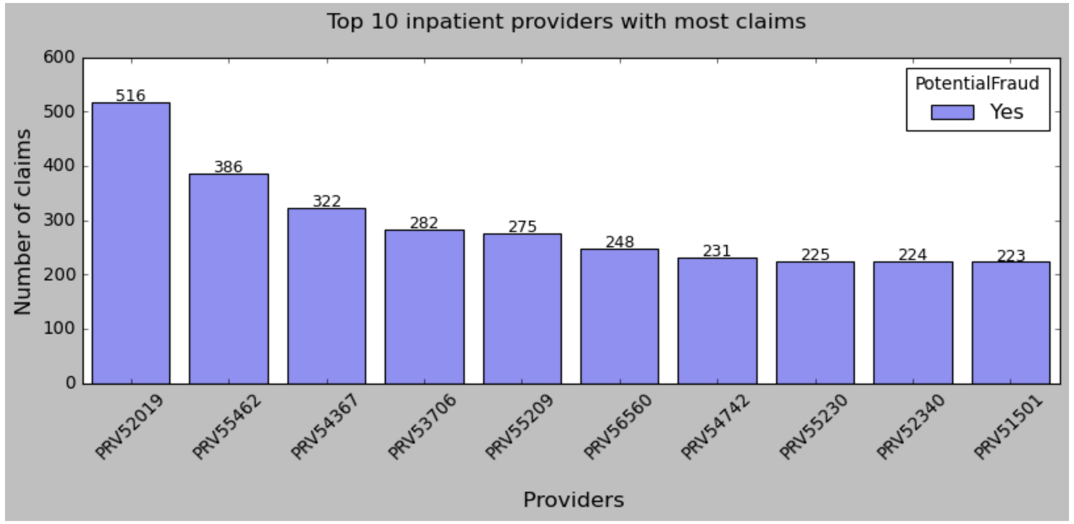


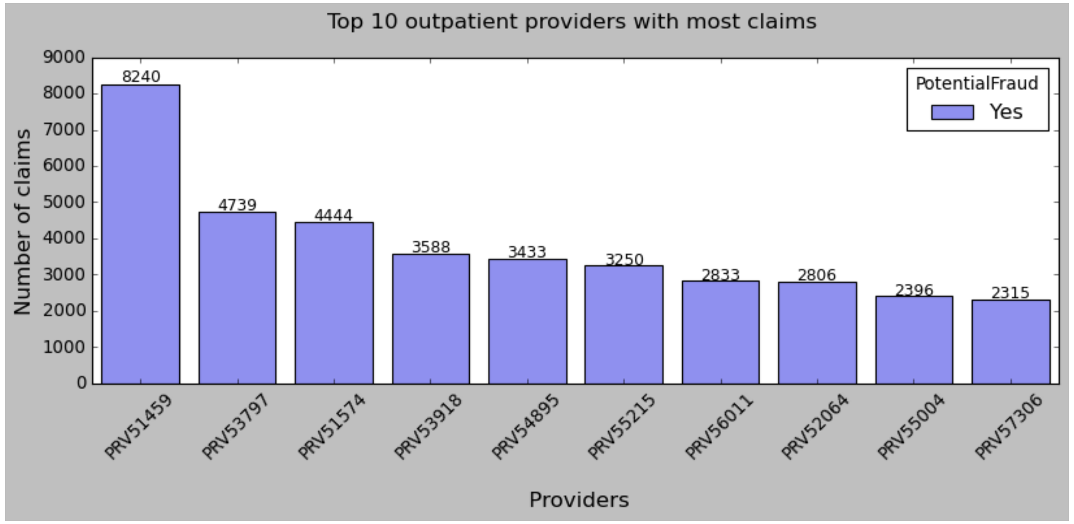




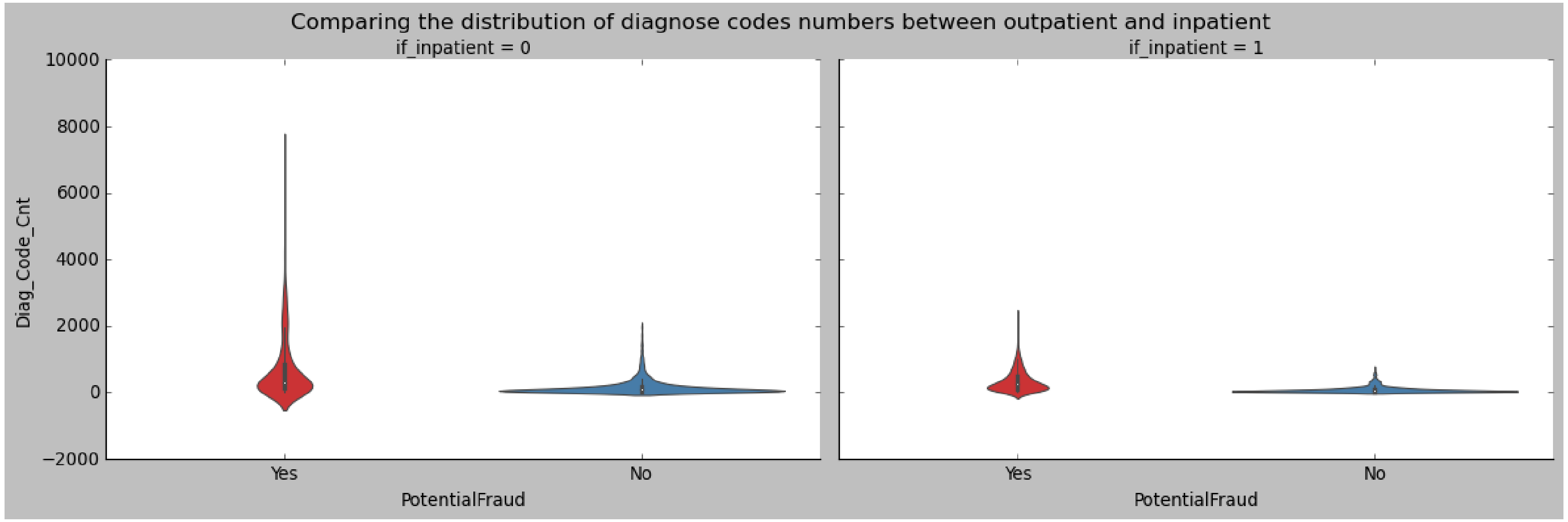
Fraud vs non-fraud provider study (continue)

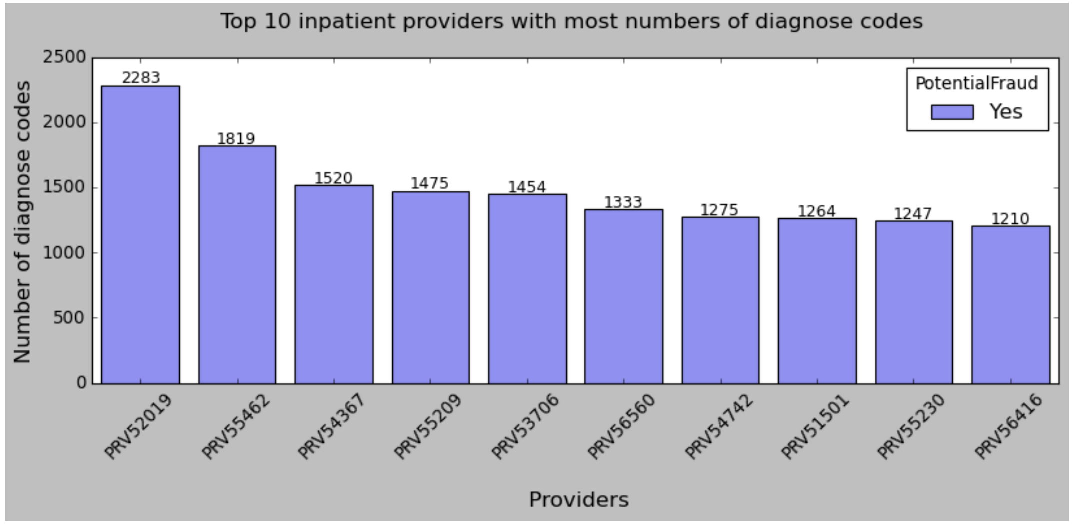


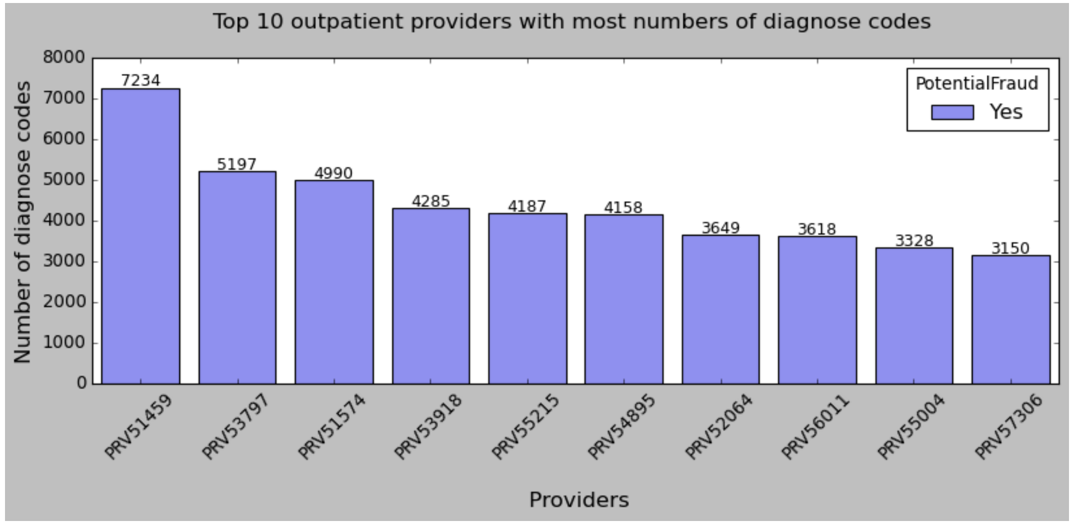




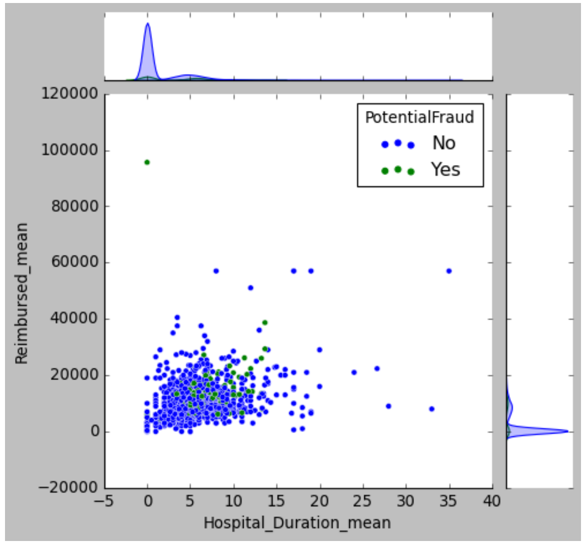
Fraud vs non-fraud provider study (continue)

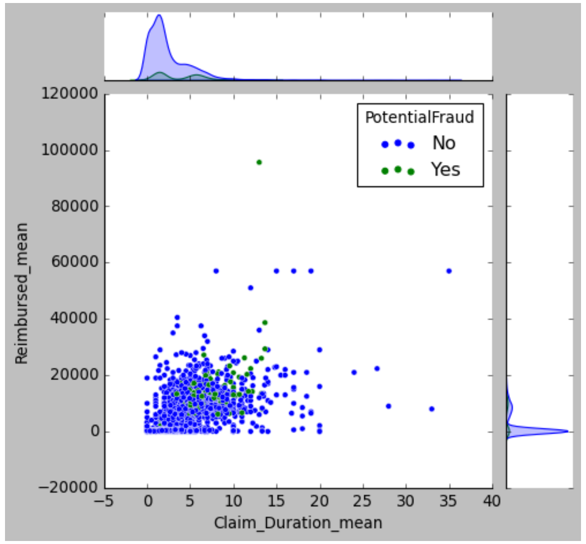


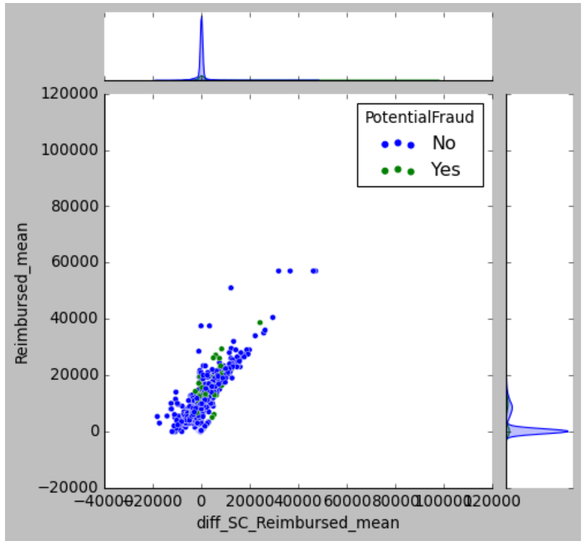


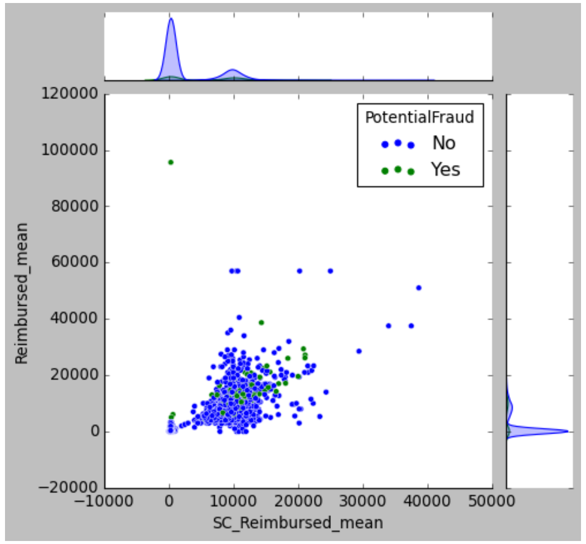


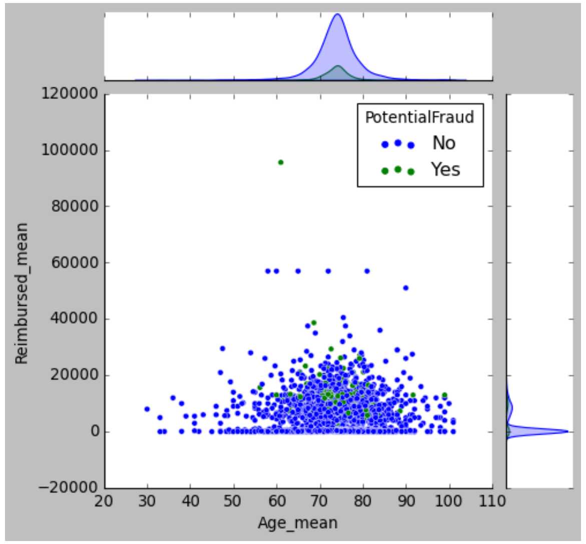
Fraud vs non-fraud provider study (continue)

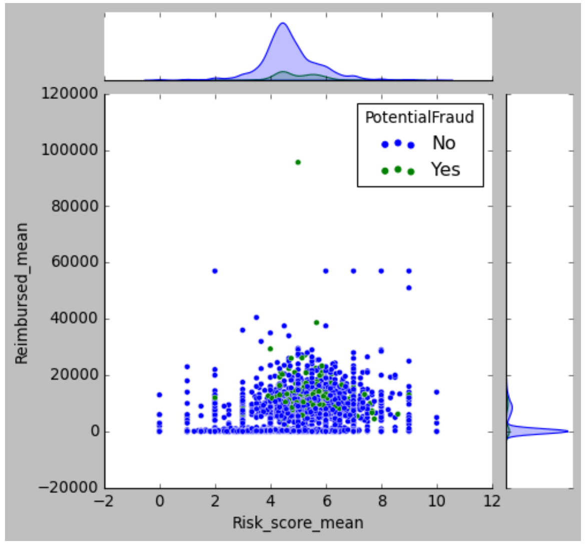




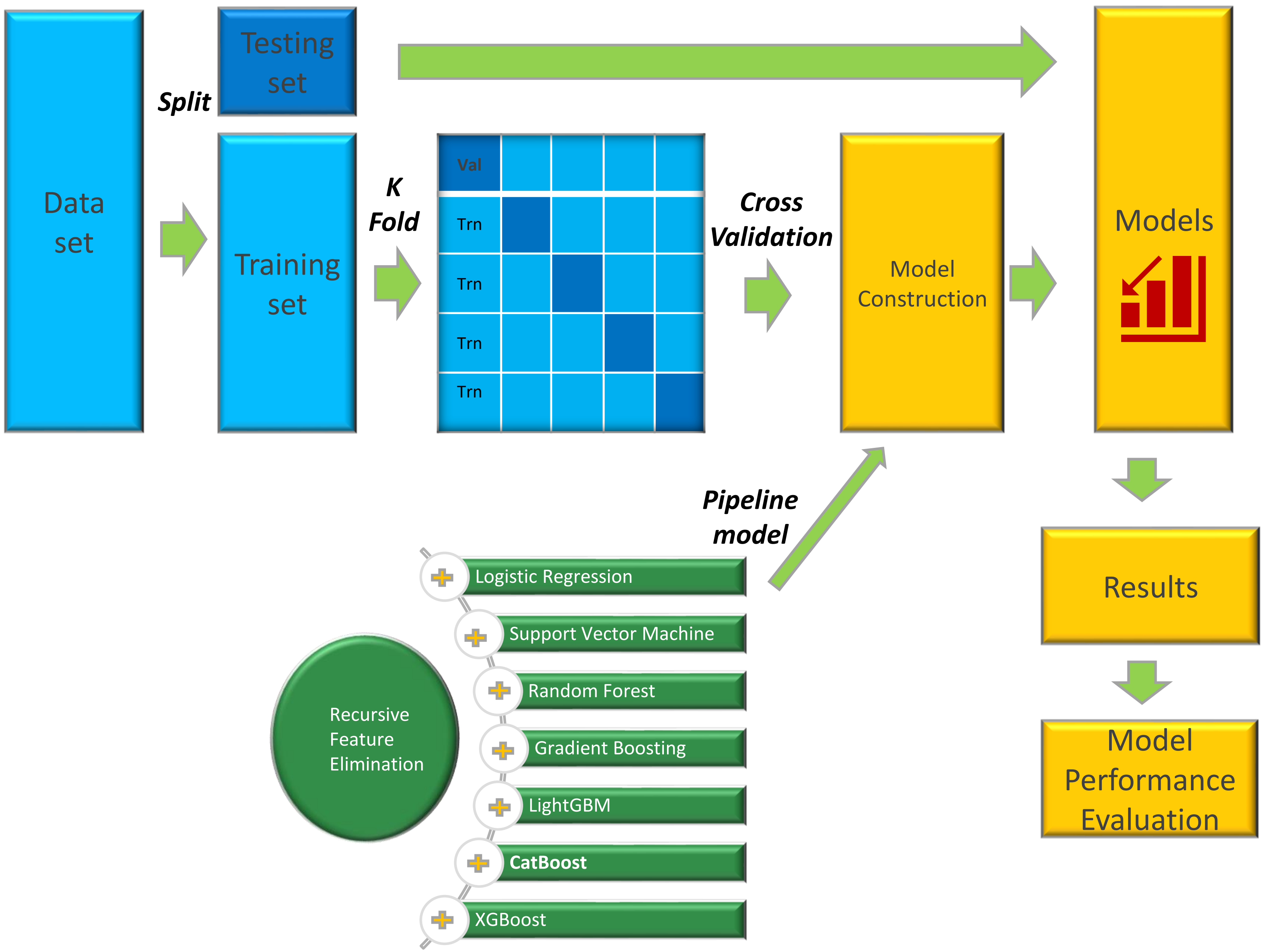


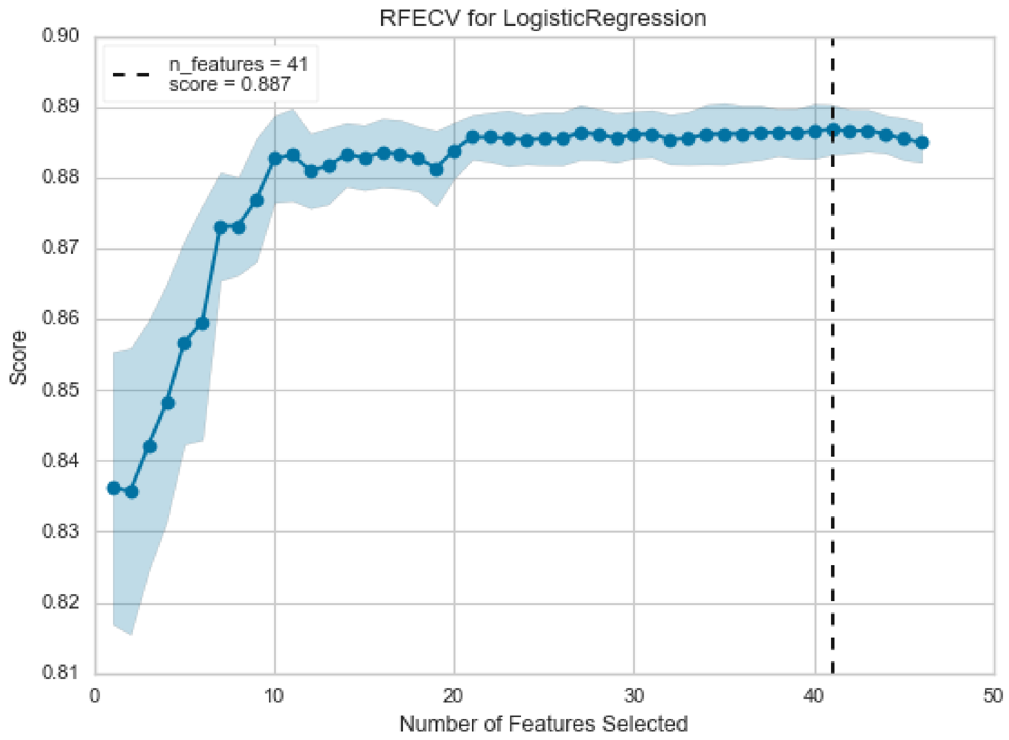


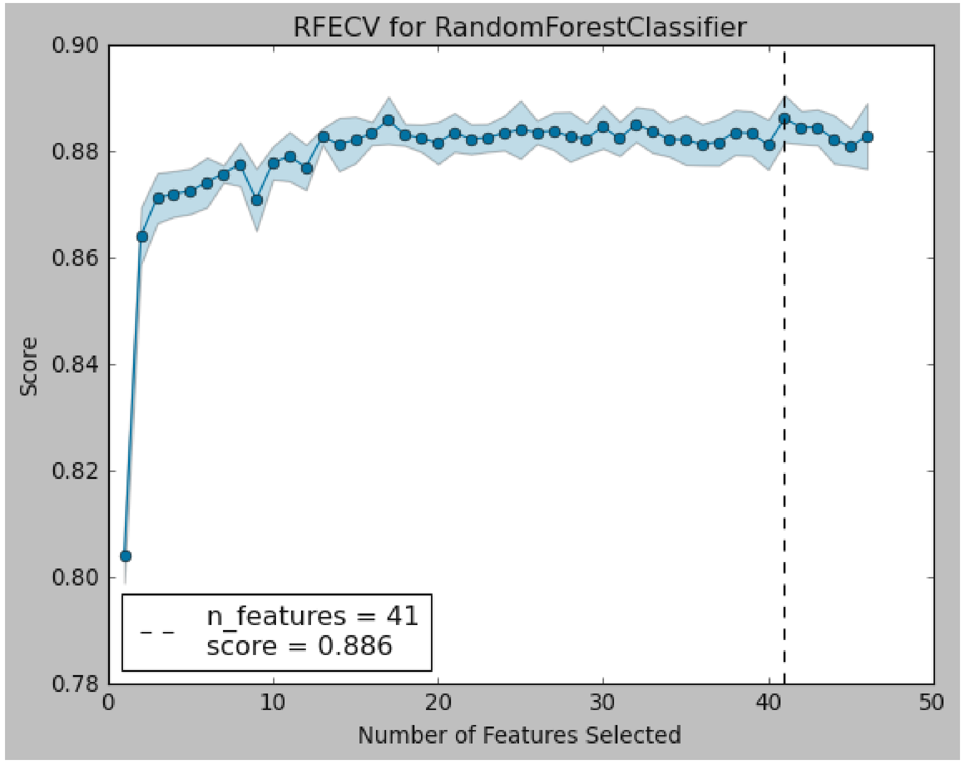




Data Modeling







Feature Selection

```python
# get a list of models to evaluate
def get_models():
    models = dict()
    # lr
    # create pipeline # Automatically selecting the number of features that resulted in the best mean score.
    rfe = RFE(estimator=LogisticRegression(), n_features_to_select=41)
    model = LogisticRegression()
    models[ 'lr' ] = Pipeline(steps=[('s',rfe),('m' ,model)])
    # svc-linear
    rfe = RFE(estimator=SVC(kernel="linear"), n_features_to_select=41 )
    model = SVC(kernel="linear")
    models[ 'svc_linear' ] = Pipeline(steps=[( 's' ,rfe),('m' ,model)])
    # rf
    rfe = RFE(estimator=RandomForestClassifier(), n_features_to_select=41)
    model = RandomForestClassifier()
    models[ 'rf' ] = Pipeline(steps=[('s' ,rfe),('m' ,model)])
    # gb
    rfe = RFE(estimator=GradientBoostingClassifier(), n_features_to_select=41)
    model = GradientBoostingClassifier()
    models[ 'gb' ] = Pipeline(steps=[( 's',rfe),('m' ,model)])
    # xgb
    rfe = RFE(estimator=XGBClassifier(), n_features_to_select=41)
    model = XGBClassifier()
    models[ 'xgb' ] = Pipeline(steps=[('s',rfe),('m' ,model)])
    # adaboost
    rfe = RFE(estimator=AdaBoostClassifier(), n_features_to_select=41)
    model = AdaBoostClassifier()
    models[ 'catb' ] = Pipeline(steps=[('s' ,rfe),('m' ,model)])
    # lgb
    rfe = RFE(estimator=LGBMClassifier(), n_features_to_select=41)
    model = LGBMClassifier()
    models[ 'lgb' ] = Pipeline(steps=[('s' ,rfe),('m' ,model)])
    return models
```

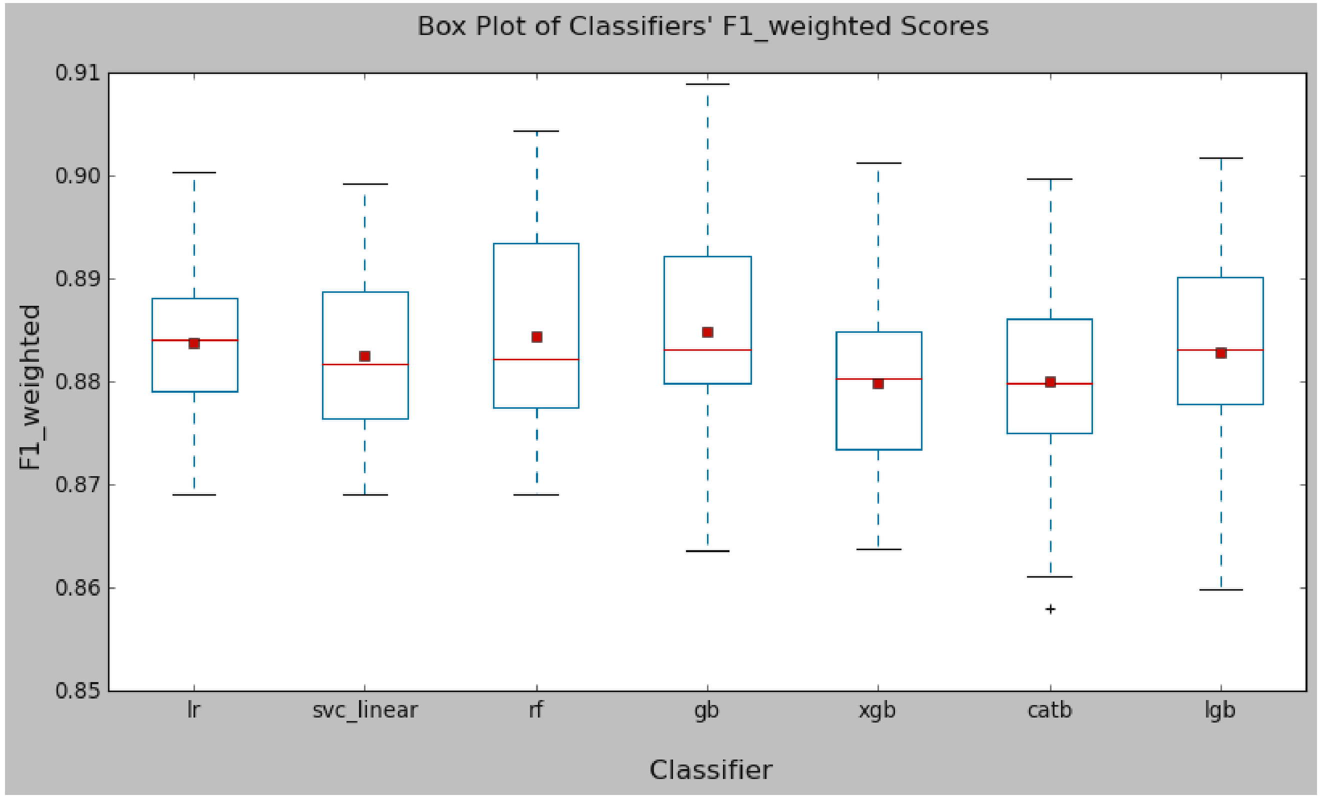
```python
# evaluate a given model using cross-validation
from sklearn.model_selection import RepeatedStratifiedKFold
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X, y, scoring= 'f1_weighted' , cv=cv, n_jobs=-1, error_score='raise')
    return scores

# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()
for name, model in models.items():
    scores = evaluate_model(model, X_train_orig, Y_train_orig)
    results.append(scores)
    names.append(name)
    print( '>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

```
>lr 0.884 (0.008)
>svc_linear 0.883 (0.008)
>rf 0.885 (0.010)
>gb 0.885 (0.010)
>xgb 0.880 (0.009)
>catb 0.880 (0.010)
>lgb 0.883 (0.010)
```

Explore Base Algorithm

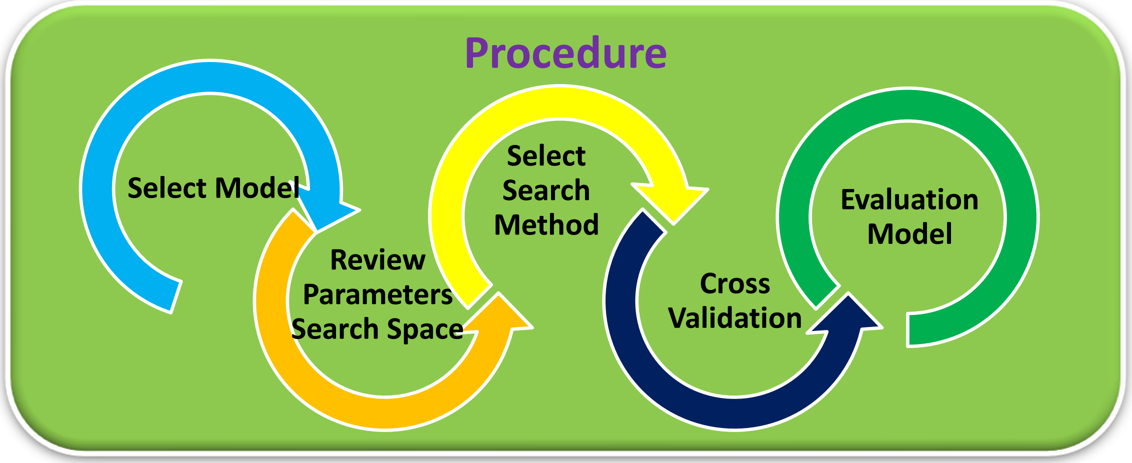
Base Algorithm comparison

- Logistic Regression, Support Vector Machine, and Random Forest classifiers generally perform well. While the mean performance of the Gradient Boosting classifier appears good, its F1_weighted score has a relatively larger variance compared to the others. This may result in less stable results.



Optimization Models

Hyperparameter Tuning



Evaluation Metrics for Imbalanced Data

- **Precision:** When **minimizing false positives** is the focus.
- **Recall:** When **minimizing false negatives** is the focus.
- **F1:** Express both concerns with a single score

Improve Recall to Minizine False Non-Fraud Provider

- SMOTE
- SMOTE+ENN

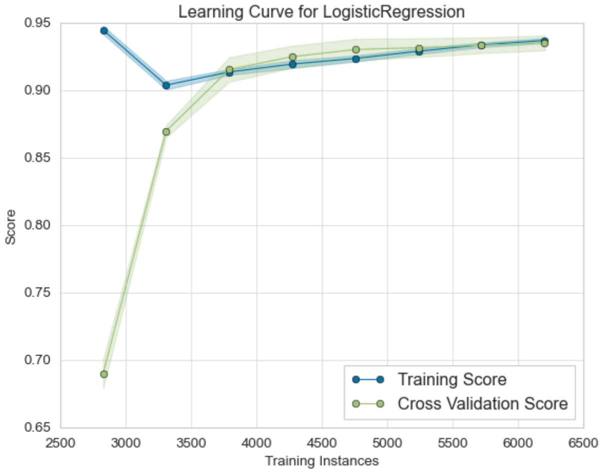
Hyperparameter Tuning

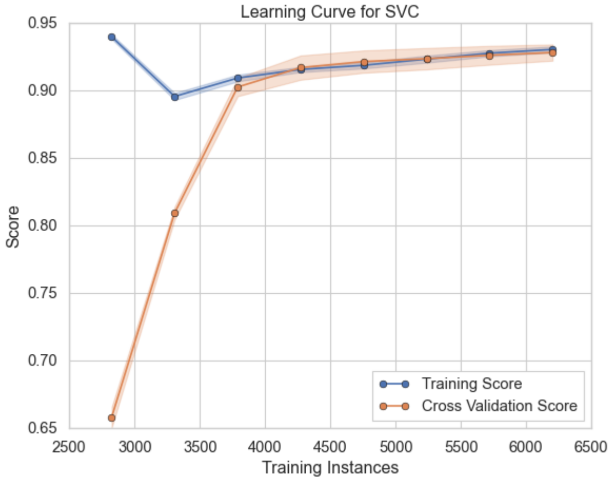
```python
# Grid Search Cross validation
# Find the best hyperparameters
def modelselection(model, parameters, scoring, cv, X_train, y_train):
    clf = GridSearchCV(estimator=model,
                       param_grid=parameters,
                       scoring= scoring,
                       cv=cv,
                       n_jobs=-1)
    # n_jobs refers to the number of CPU's that you want to use for excution, -1 means that use all available computing power.
    clf.fit(X_train, y_train)
    cv_results = clf.cv_results_
    best_parameters = clf.best_params_
    best_result = clf.best_score_
    print('The best parameters for classifier is', best_parameters)
    print('The best training score is %.3f:'% best_result)
    #  print(sorted(cv_results.keys()))
    return cv_results, best_parameters, best_result
```

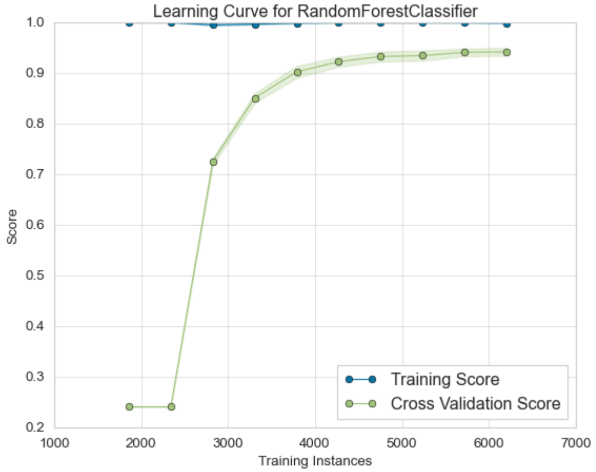
```python
model_rf = RandomForestClassifier(random_state=42)

paras_rf={'n_estimators':[30,50,100],
          'max_depth':[i for i in range(5,16,2)], # Minimum number of samples to consider to split a node:
          'min_samples_split':[2, 5, 10, 15, 20]} # Minimum number of samples to consider at each leaf node:

cv_results, best_param, best_result = modelselection(model_rf, paras_rf, scoring, cv, X_train_orig_rfe, Y_train_orig)
```

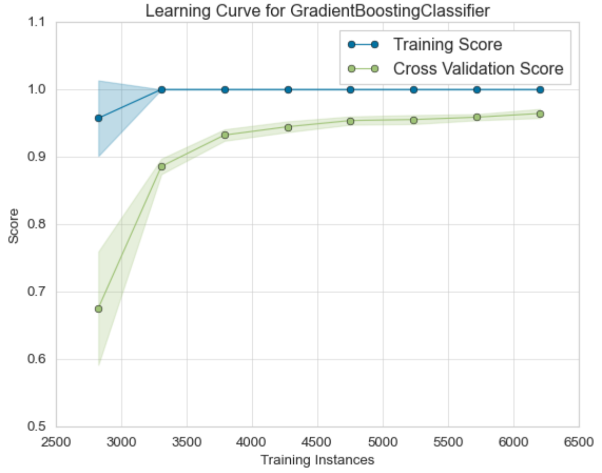
```
The best parameters for classifier is {'max_depth': 9, 'min_samples_split': 15, 'n_estimators': 50}
The best training score is 0.887:
```

Learning Curve

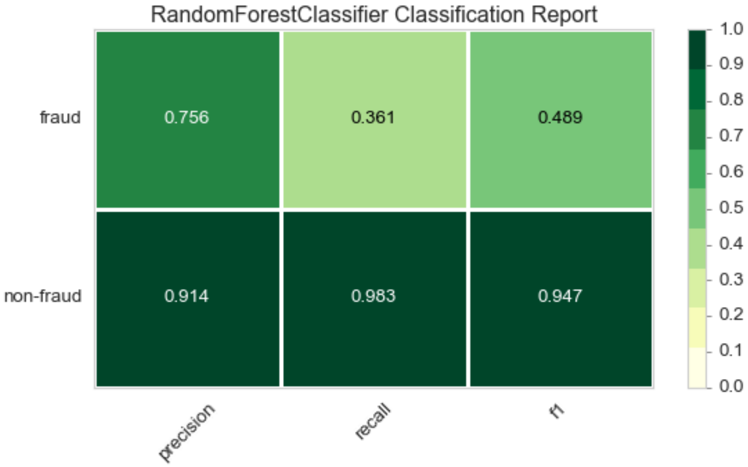


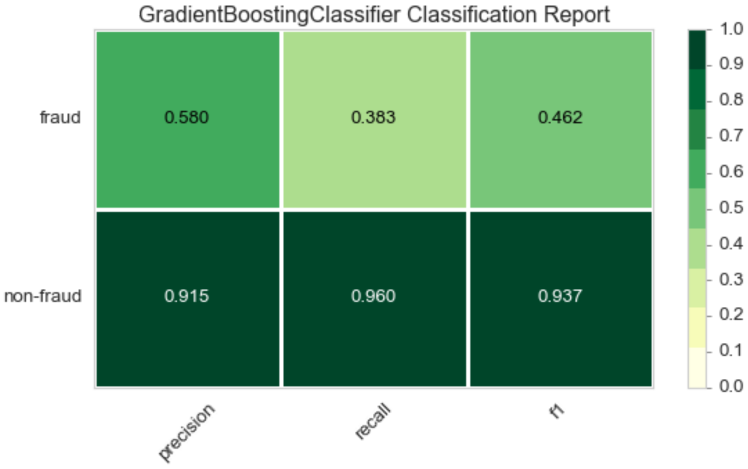


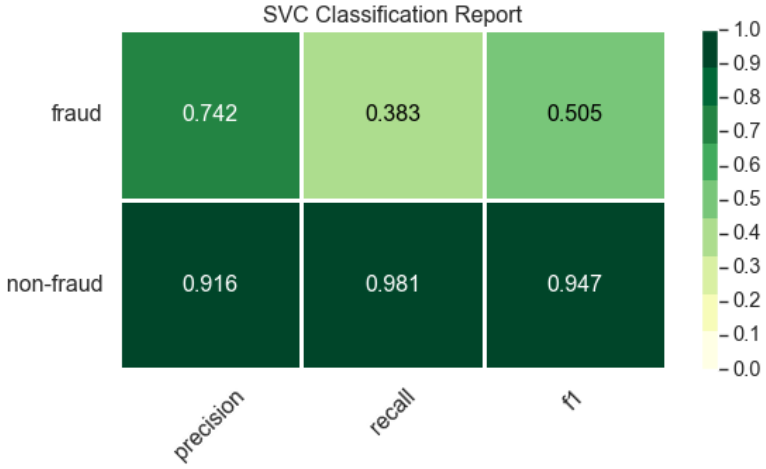




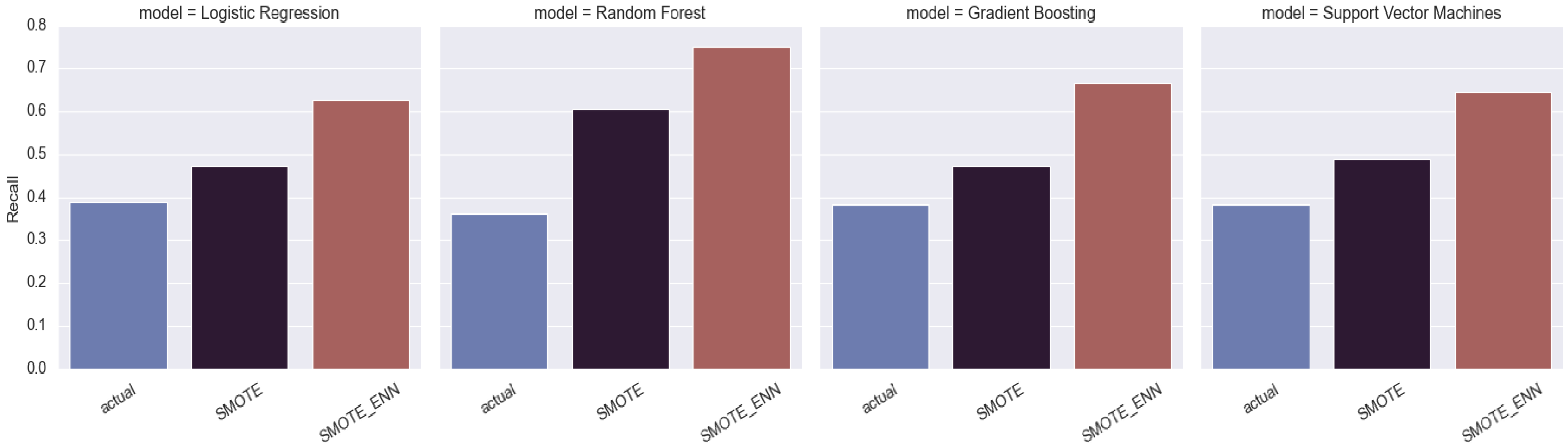
Precision, Recall, F1-score for Models





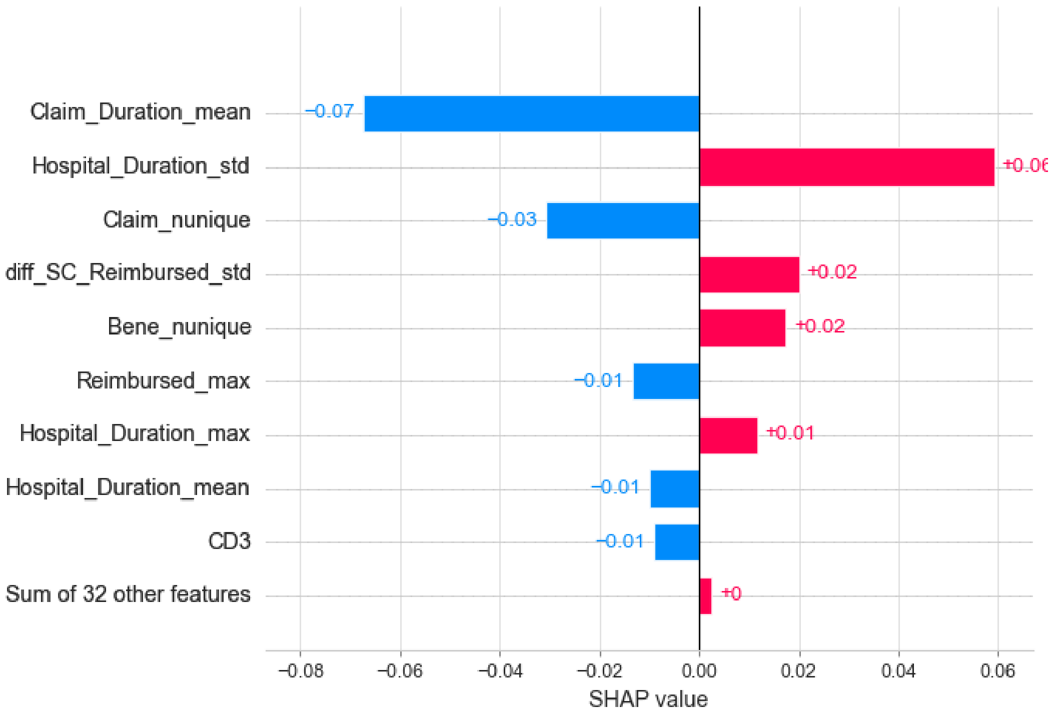


Maximize Minority's Recall by using SMOTE Techniques

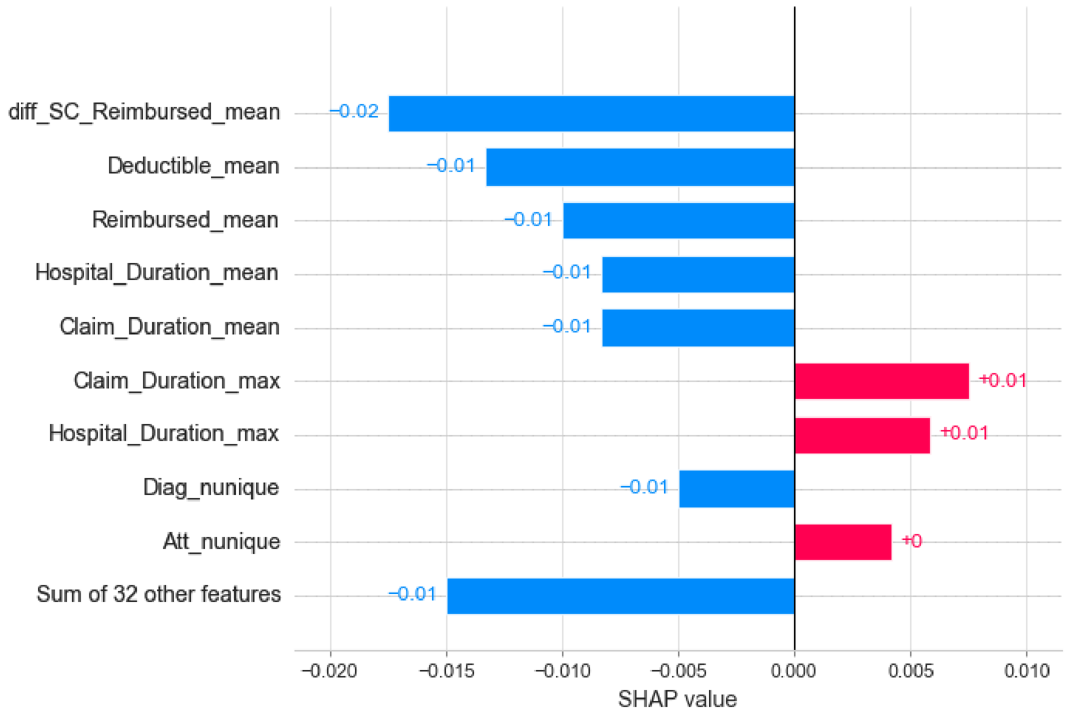


Feature importance: SHAP value for class 0

Logistic Regression Model:

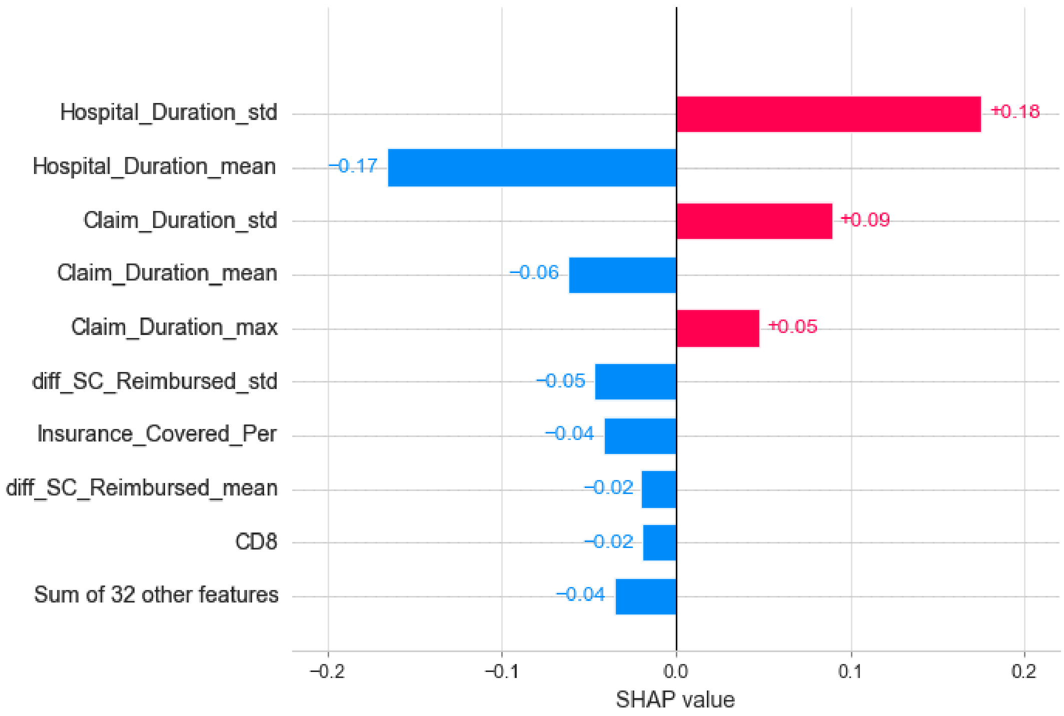


Random Forest Model:

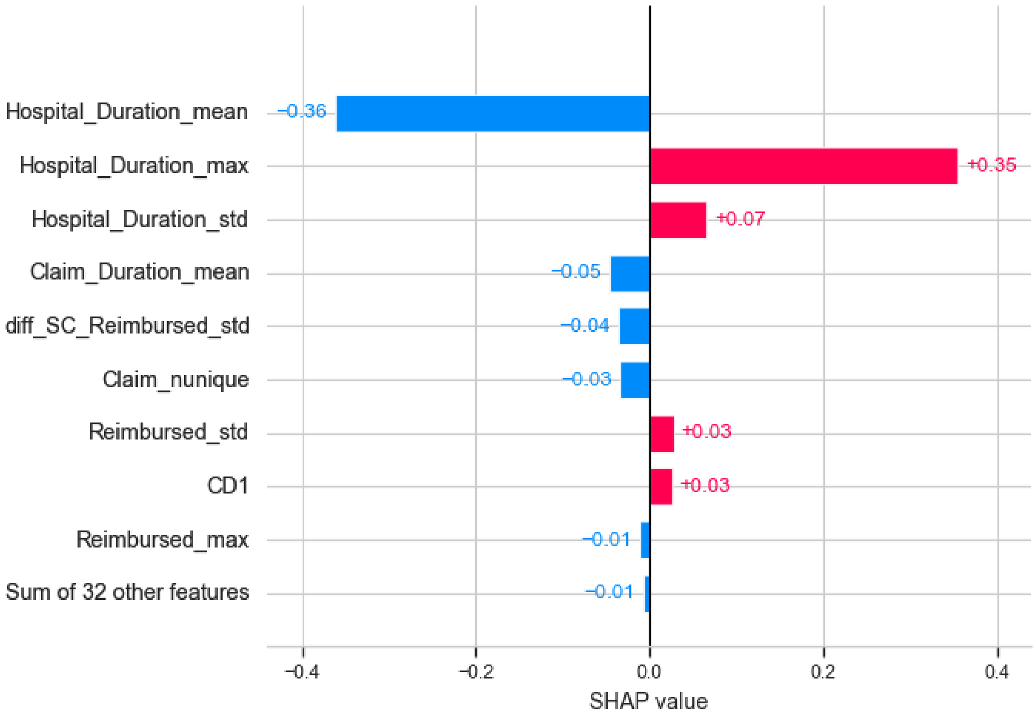


Feature importance:

Gradient Boosting Model:



Linear SVM Model:



Conclusion

Possibly Fraud Providers	**Non-Fraud Providers**
High average claim settlement time with small variance	Low average claim settlement time with large variance
High average hospital duration time with small variance	Low average hospital duration time with large variance
High number of patient insurance claims	Low number of patient insurance claims
High average reimbursement	Low average reimbursement
High average deductible	Low average deductible
High difference of claim reimbursement from county state mean	Low difference of claim reimbursement from county state mea
Low number of Beneficiaries	High number of Beneficiaries
High number of diagnosis codes listed on claims	Low number of diagnosis codes listed on claims
Low number of Physician	High number of Physician

Reference

- National Health Expenditure Data from https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/NationalHealthExpendData/NHE-Fact-Sheet
- Healthcare Provider Fraud Data from https://www.kaggle.com/datasets/rohitrox/healthcare-provider-fraud-detection-analysis