homework9

November 13, 2019

NAME: Zhi Li
SECTION: 113523595
CS 5970: Machine Learning Practices

1 Homework 9: Decision

1.1 Assignment Overview

Follow the TODOs and read through and understand any provided code. Post any questions you might have to the Canvas discussion. For all plots, make sure all necessary axes and curves are clearly and accurately labeled. Include figure/plot titles appropriately as well.

1.1.1 Task

For this assignment you will be exploring Decision Tree Classifiers.

1.1.2 Data set

The data file can be found on Canvas under Files/Homework Solutions, and on git and the server under datasets/fraud_detection/health_provider_fraud.csv.

These data were re-configured from a dataset collected for the purpose of detecting Health care Provider Fraud. Total Medicare spending increases exponentially due to frauds in Medicare claims. Healthcare fraud involves health care providers, physicians, patients, and beneficiaries acting in-tandum to construct fraudulent claims.

The goal is to "predict potentially fraudulent providers" from summary statistics of their filed healthcare claims.

Features
The features are aggregate statistics computed as either the mean or the sum. For the following features, the column is indicative of the average value for the provider's claims:
* InscClaimAmtReimbursed
* DeductibleAmtPaid * NoOfMonths_PartACov * NoOfMonths_PartBCov * IPAnnualReimbursementAmt * IPAnnualDeductibleAmt * OPAnnualReimbursementAmt * OPAnnualDeductibleAmt * NumPhysiciansSeen * NumProcedures * NumDiagnosisClaims * Age

For the following features, the column is indicative of the total number among the provider's claims:
* ChronicCond_Alzheimer
* ChronicCond_Heartfailure
* ChronicCond_KidneyDisease
* ChronicCond_Cancer
* ChronicCond_ObstrPulmonary
* ChronicCond_Depression
* ChronicCond_Diabetes
* ChronicCond_IschemicHeart
* ChronicCond_Osteoporasis
* ChronicCond_rheumatoidarthritis
* ChronicCond_stroke
* RenalDiseaseIndicator

These data were amalagmated from the HEALTHCARE PROVIDER FRAUD DETECTION ANALYSIS data set on Kaggle.

1.1.3 Objectives

- Introduction to Decision Trees

1.1.4 Notes

- Do not save work within the ml_practices folder

1.1.5 General References

- Guide to Jupyter
- Python Built-in Functions
- Python Data Structures
- Numpy Reference
- Numpy Cheat Sheet
- Summary of matplotlib
- DataCamp: Matplotlib
- Pandas DataFrames
- Sci-kit Learn Linear Models
- Sci-kit Learn Ensemble Models
- Sci-kit Learn Metrics
- Sci-kit Learn Model Selection
- Sci-kit Learn Pipelines
- Sci-kit Learn Preprocessing

[2]:
```python
# THESE FIRST 3 IMPORTS ARE FROM FILES IN THE ML_PRACTICES FOLDER UNDER HW9
# Use the versions found in the hw9 folder as some changes were made
import visualize
import metrics_plots
```

```python
from pipeline_components import DataSampleDropper, DataFrameSelector, DataScaler

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re, os, pathlib
import time as timelib

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix, roc_curve, auc
from sklearn.metrics import log_loss, f1_score
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder
from sklearn.externals import joblib
import pickle as pkl

FIGW = 5
FIGH = 5
FONTSIZE = 12

plt.rcParams['figure.figsize'] = (FIGW, FIGH)
plt.rcParams['font.size'] = FONTSIZE

plt.rcParams['xtick.labelsize'] = FONTSIZE
plt.rcParams['ytick.labelsize'] = FONTSIZE

%matplotlib inline
plt.style.use('ggplot')
```

[3]:
```python
""" PROVIDED
Display current working directory of this notebook. If you are using
relative paths for your data, then it needs to be relative to the CWD.
"""
HOME_DIR = pathlib.Path.home()
pathlib.Path.cwd()
```

[3]: PosixPath('/home/jovyan')

2 LOAD DATA

[4]:
```
# TODO: set path appropriately.
# data file can be found on canvas under Files/Homework Solutions, and on git
# and the server under datasets/fraud_detection/
fname = "ml_practices/imports/datasets/fraud_detection/health_provider_fraud.
↪csv"
claims_data = pd.read_csv(fname)
claims_data.shape
```

[4]:
```
(5410, 25)
```

[5]:
```
""" PROVIDED
Display data info
"""
claims_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5410 entries, 0 to 5409
Data columns (total 25 columns):
Provider                          5410 non-null object
PotentialFraud                    5410 non-null bool
Age                               5410 non-null float64
NumPhysiciansSeen                 5410 non-null float64
NumProcedures                     5410 non-null float64
NumDiagnosisClaims                5410 non-null float64
InscClaimAmtReimbursed            5410 non-null float64
DeductibleAmtPaid                 5409 non-null float64
NoOfMonths_PartACov               5410 non-null float64
NoOfMonths_PartBCov               5410 non-null float64
IPAnnualReimbursementAmt          5410 non-null float64
IPAnnualDeductibleAmt             5410 non-null float64
OPAnnualReimbursementAmt          5410 non-null float64
OPAnnualDeductibleAmt             5410 non-null float64
ChronicCond_Alzheimer             5410 non-null int64
ChronicCond_Heartfailure          5410 non-null int64
ChronicCond_KidneyDisease         5410 non-null int64
ChronicCond_Cancer                5410 non-null int64
ChronicCond_ObstrPulmonary        5410 non-null int64
ChronicCond_Depression            5410 non-null int64
ChronicCond_Diabetes              5410 non-null int64
ChronicCond_IschemicHeart         5410 non-null int64
ChronicCond_Osteoporasis          5410 non-null int64
ChronicCond_rheumatoidarthritis   5410 non-null int64
ChronicCond_stroke                5410 non-null int64
dtypes: bool(1), float64(12), int64(11), object(1)
memory usage: 1019.7+ KB
```

[6]:
```python
""" PROVIDED
Display the head of the data
"""
claims_data.head()
```

[6]:
```
   Provider  PotentialFraud        Age  NumPhysiciansSeen  NumProcedures  \
0  PRV51001           False  78.840000           1.280000       0.120000
1  PRV51003            True  70.022727           1.181818       0.363636
2  PRV51004           False  72.161074           1.322148       0.000000
3  PRV51005            True  70.475536           1.209442       0.000000
4  PRV51007           False  69.291667           1.125000       0.013889

   NumDiagnosisClaims  InscClaimAmtReimbursed  DeductibleAmtPaid  \
0            3.640000             4185.600000         213.600000
1            5.765152             4588.409091         502.166667
2            2.751678              350.134228           2.080537
3            2.786266              241.124464           3.175966
4            3.208333              468.194444          45.333333

   NoOfMonths_PartACov  NoOfMonths_PartBCov  …  ChronicCond_Heartfailure  \
0            12.000000            12.000000  …                        19
1            11.818182            11.871212  …                        80
2            11.865772            11.959732  …                        88
3            11.907296            11.939914  …                       680
4            11.833333            11.833333  …                        40

   ChronicCond_KidneyDisease  ChronicCond_Cancer  ChronicCond_ObstrPulmonary  \
0                         17                   5                          10
1                         64                  10                          41
2                         50                  16                          41
3                        507                 165                         295
4                         22                  12                          16

   ChronicCond_Depression  ChronicCond_Diabetes  ChronicCond_IschemicHeart  \
0                       9                    21                         23
1                      54                   100                        112
2                      63                   105                        108
3                     485                   799                        895
4                      29                    49                         51

   ChronicCond_Osteoporasis  ChronicCond_rheumatoidarthritis  \
0                         6                                8
1                        33                               38
2                        49                               46
3                       344                              331
4                        21                               22
```

```
   ChronicCond_stroke
0                   6
1                  12
2                  17
3                 124
4                  12

[5 rows x 25 columns]
```

[7]:
```
""" PROVIDED
Display the summary statistics
Make sure you skim this
"""
claims_data.describe()
```

[7]:
```
               Age  NumPhysiciansSeen  NumProcedures  NumDiagnosisClaims  \
count  5410.000000        5410.000000    5410.000000         5410.000000
mean     73.731027           1.227410       0.108011            3.676631
std       4.712307           0.220822       0.246305            1.882603
min      34.000000           0.500000       0.000000            0.000000
25%      71.768368           1.000000       0.000000            2.696134
50%      73.863636           1.200000       0.000000            3.000000
75%      75.760000           1.375000       0.083333            3.847902
max     101.000000           3.000000       3.000000           11.000000

       InscClaimAmtReimbursed  DeductibleAmtPaid  NoOfMonths_PartACov  \
count             5410.000000        5409.000000          5410.000000
mean              1740.679369         155.643175            11.919716
std               3484.473124         306.489453             0.395682
min                  0.000000           0.000000             0.000000
25%                232.394593           0.312500            11.994207
50%                356.085106           4.285714            12.000000
75%               1490.154301         137.418605            12.000000
max              57000.000000        1068.000000            12.000000

       NoOfMonths_PartBCov  IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
count          5410.000000               5410.000000            5410.000000
mean             11.930647               6166.692586             666.980865
std               0.310612               6203.422910             623.108956
min               0.000000                  0.000000               0.000000
25%              11.965836               2902.238095             356.000000
50%              12.000000               4729.047927             527.580008
75%              12.000000               7336.173195             801.000000
max              12.000000             103000.000000           12068.000000

       …  ChronicCond_Heartfailure  ChronicCond_KidneyDisease  \
count  …               5410.000000                5410.000000
```

```
mean   …          60.921072            42.510906
std    …         158.698296           110.048136
min    …           0.000000             0.000000
25%    …           6.000000             4.000000
50%    …          18.000000            13.000000
75%    …          52.750000            37.000000
max    …        4638.000000          3111.000000

       ChronicCond_Cancer  ChronicCond_ObstrPulmonary  ChronicCond_Depression  \
count         5410.000000                 5410.000000             5410.000000
mean            15.620148                   32.288540               44.863956
std             41.558020                   82.958866              117.563035
min              0.000000                    0.000000                0.000000
25%              1.000000                    3.000000                4.000000
50%              5.000000                   10.000000               13.000000
75%             13.000000                   29.000000               39.000000
max           1238.000000                 2312.000000             3592.000000

       ChronicCond_Diabetes  ChronicCond_IschemicHeart  \
count           5410.000000                5410.000000
mean              72.783549                  78.341959
std              190.919202                 205.233787
min                0.000000                   0.000000
25%                7.000000                   7.000000
50%               22.000000                  23.000000
75%               62.750000                  67.000000
max             5784.000000                6074.000000

       ChronicCond_Osteoporasis  ChronicCond_rheumatoidarthritis  \
count               5410.000000                      5410.000000
mean                  32.775231                        32.107024
std                   85.862305                        84.497824
min                    0.000000                         0.000000
25%                    3.000000                         3.000000
50%                   10.000000                         9.000000
75%                   28.000000                        28.000000
max                 2531.000000                      2511.000000

       ChronicCond_stroke
count         5410.000000
mean            10.495564
std             27.171512
min              0.000000
25%              1.000000
50%              3.000000
75%              9.000000
max            810.000000
```

```
[8 rows x 23 columns]
```

3 PRE-PROCESS DATA

[8]:
```python
""" PROVIDED
Construct preprocessing pipeline
"""
selected_features = claims_data.columns
scaled_features = ['InscClaimAmtReimbursed', 'DeductibleAmtPaid',
                   'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
                   'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt']

pipe = Pipeline([
    ('RowDropper', DataSampleDropper()),
    ('FeatureSelector', DataFrameSelector(selected_features)),
    ('Scale', DataScaler(scaled_features))
])
```

[9]:
```python
""" TODO
Pre-process the data using the defined pipeline
"""
processed_data = pipe.fit_transform(claims_data)# TODO
processed_data.shape
```

[9]:
```
(5409, 25)
```

[10]:
```python
""" TODO
Verify all NaNs removed
"""

processed_data.isna().any()
```

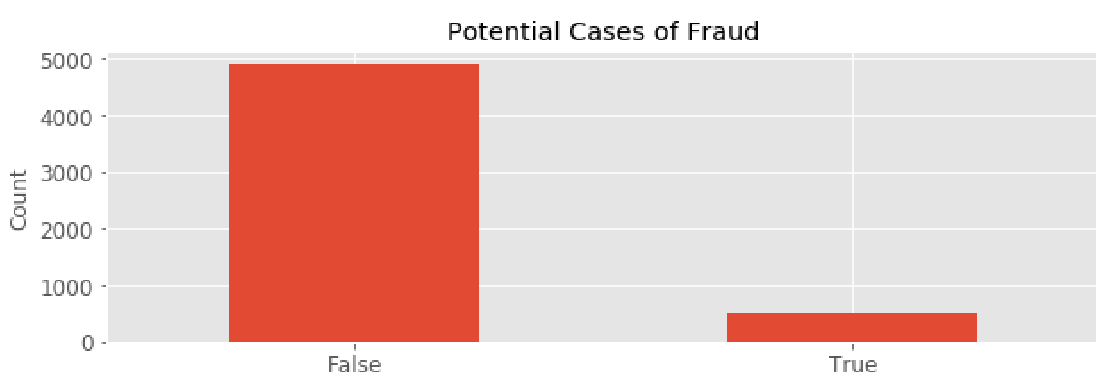
[10]:
```
Provider                          False
PotentialFraud                    False
Age                               False
NumPhysiciansSeen                 False
NumProcedures                     False
NumDiagnosisClaims                False
InscClaimAmtReimbursed            False
DeductibleAmtPaid                 False
NoOfMonths_PartACov               False
NoOfMonths_PartBCov               False
IPAnnualReimbursementAmt          False
IPAnnualDeductibleAmt             False
```

```
OPAnnualReimbursementAmt            False
OPAnnualDeductibleAmt               False
ChronicCond_Alzheimer               False
ChronicCond_Heartfailure            False
ChronicCond_KidneyDisease           False
ChronicCond_Cancer                  False
ChronicCond_ObstrPulmonary          False
ChronicCond_Depression              False
ChronicCond_Diabetes                False
ChronicCond_IschemicHeart           False
ChronicCond_Osteoporasis            False
ChronicCond_rheumatoidarthritis     False
ChronicCond_stroke                  False
dtype: bool
```

4 VISUALIZE DATA

[11]:
```
""" PROVIDED
Plot the class distributions for no potential fraud and potential fraud
"""
class_counts = pd.value_counts(processed_data['PotentialFraud'])
class_counts.plot(kind='bar', rot=0, figsize=(10,3))
plt.title("Potential Cases of Fraud")
plt.ylabel("Count")

# Display the class fractions
nsamples, nfeatures = processed_data.shape
class_counts / nsamples
```

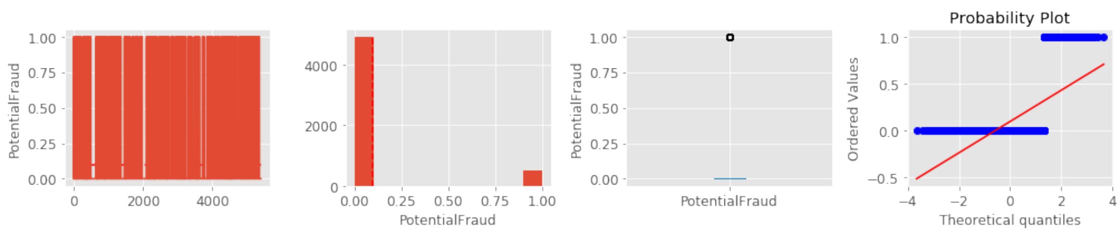
[11]:
```
False    0.906452
True     0.093548
Name: PotentialFraud, dtype: float64
```



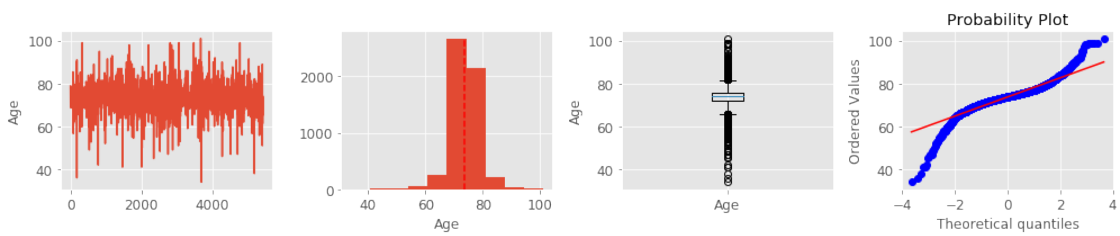
[12]:
```
""" PROVIDED
Extract positions of the postive and negative cases
"""
pos = processed_data['PotentialFraud'] == 1
neg = processed_data['PotentialFraud'] == 0
```

[13]:
```
""" PROVIDED
Visualize the data using visualize.featureplots
"""
# Drop the provider name from the visualized data since it is not numeric
cdata = processed_data.drop(['Provider'], axis=1).astype('float64')
visualize.featureplots(cdata.values, cdata.columns)
```

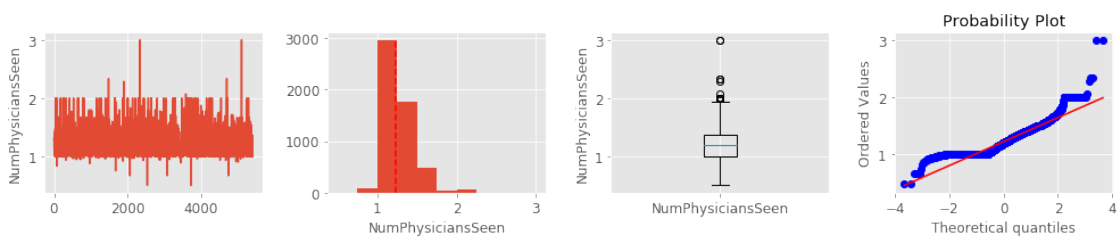
FEATURE: PotentialFraud



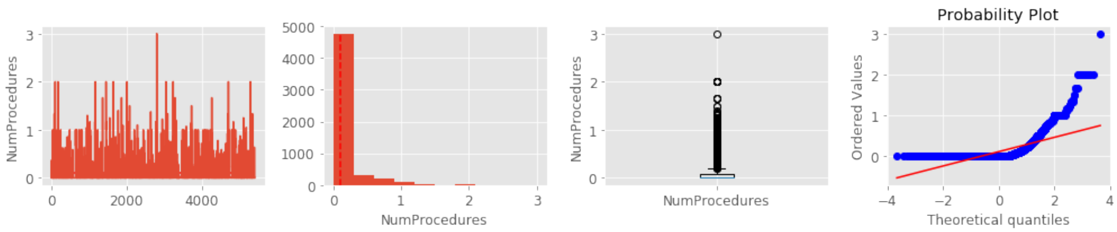
FEATURE: Age



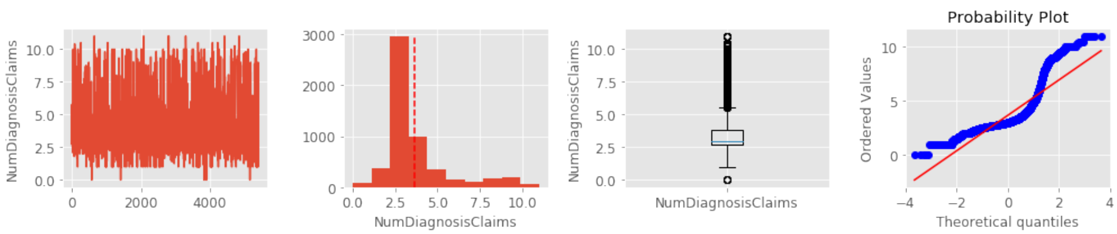
FEATURE: NumPhysiciansSeen



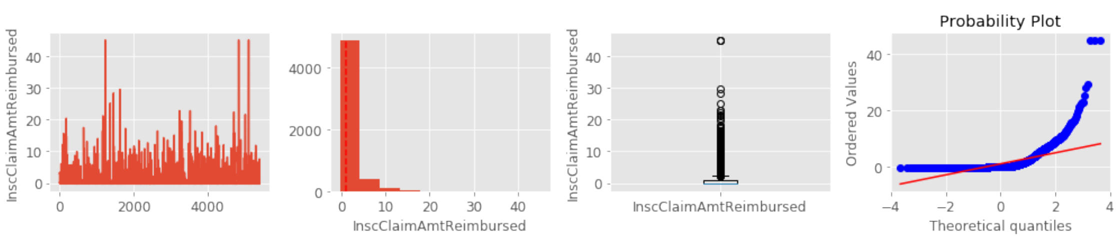
FEATURE: NumProcedures



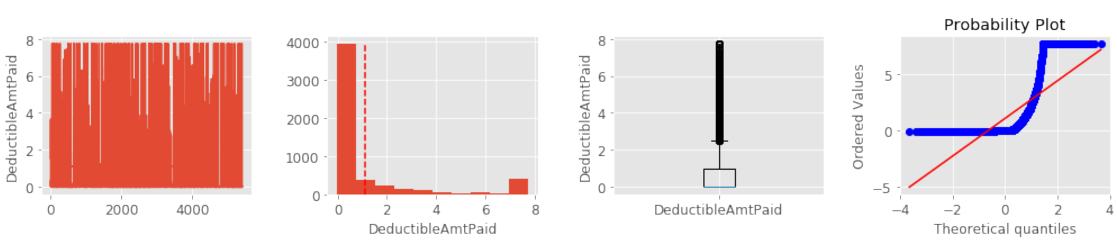
FEATURE: NumDiagnosisClaims



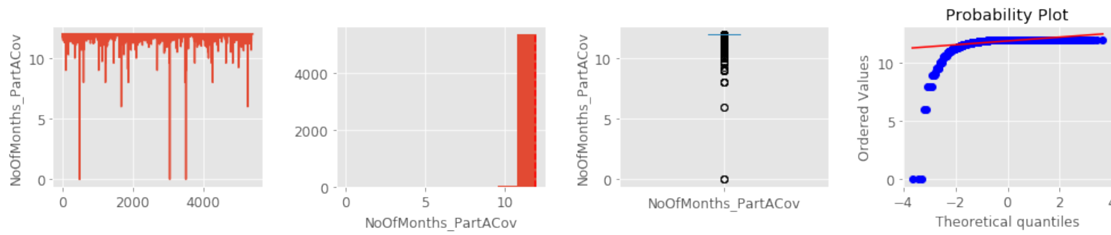
FEATURE: InscClaimAmtReimbursed



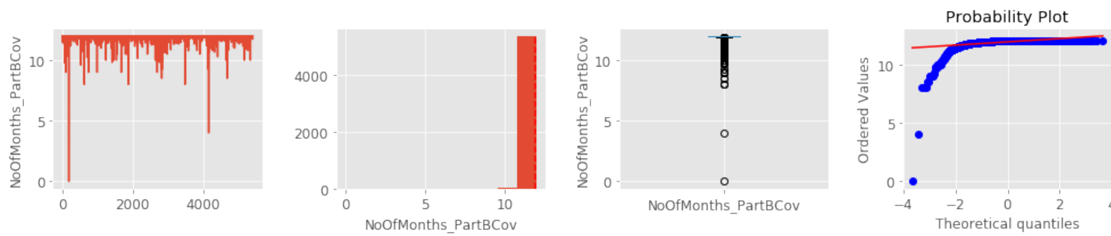
FEATURE: DeductibleAmtPaid



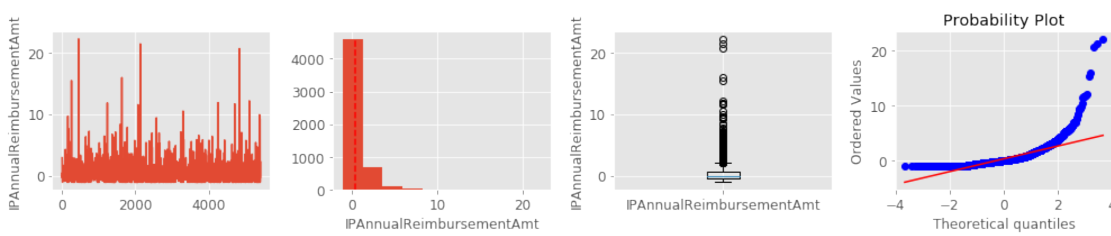
FEATURE: NoOfMonths_PartACov



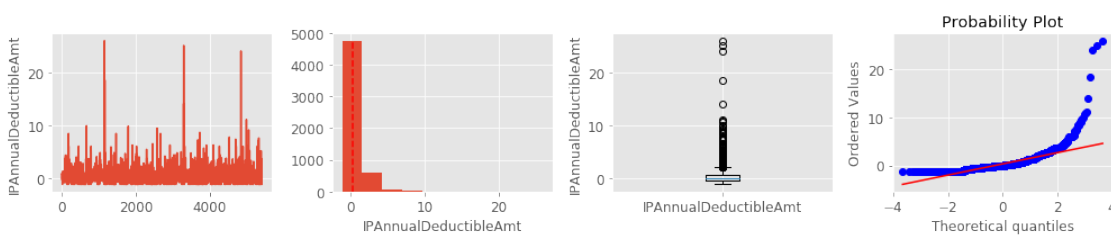
FEATURE: NoOfMonths_PartBCov



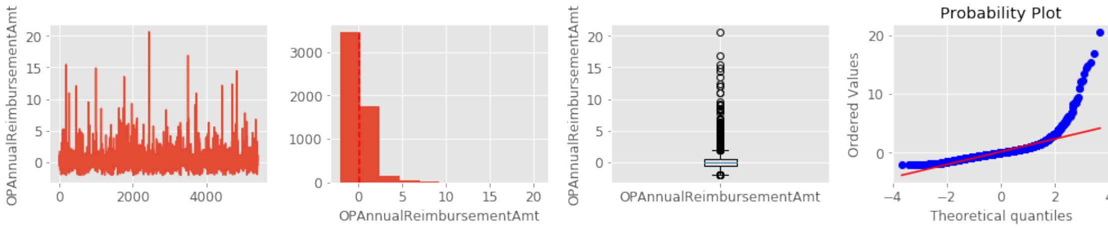
FEATURE: IPAnnualReimbursementAmt



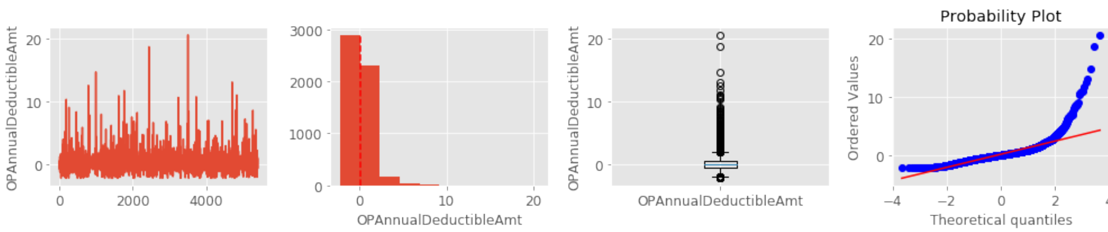
FEATURE: IPAnnualDeductibleAmt



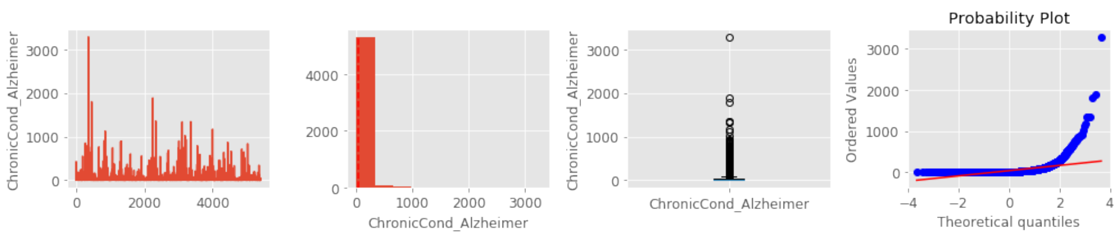
FEATURE: OPAnnualReimbursementAmt



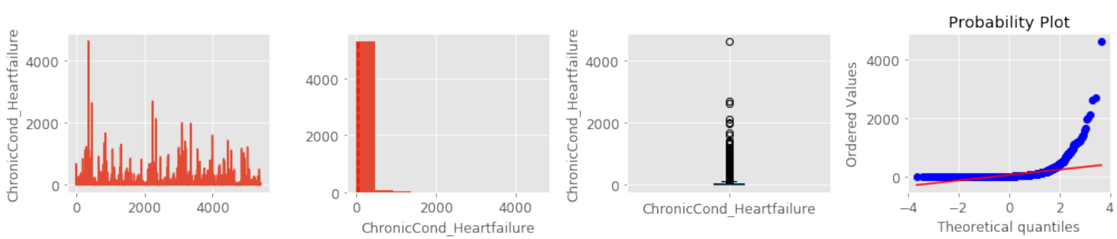
FEATURE: OPAnnualDeductibleAmt



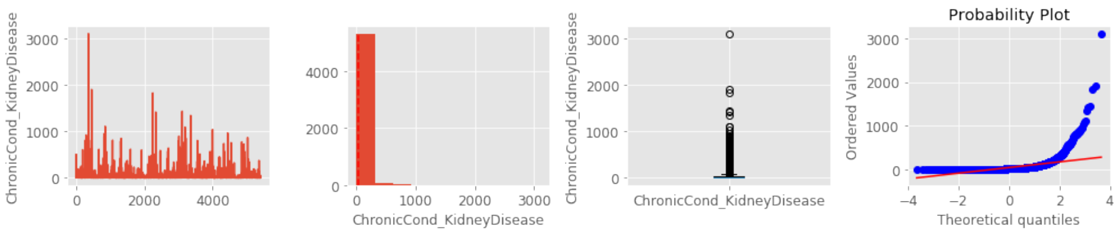
FEATURE: ChronicCond_Alzheimer



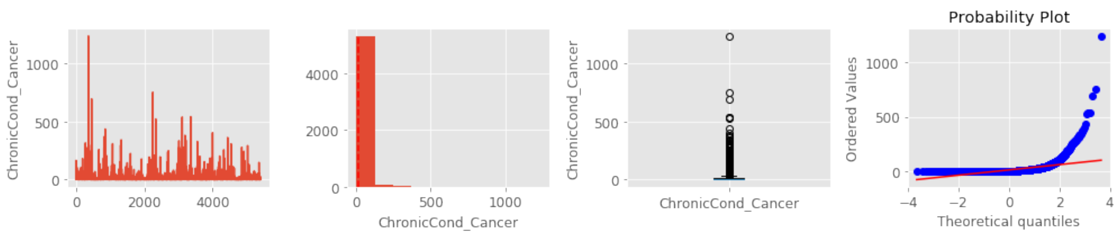
FEATURE: ChronicCond_Heartfailure



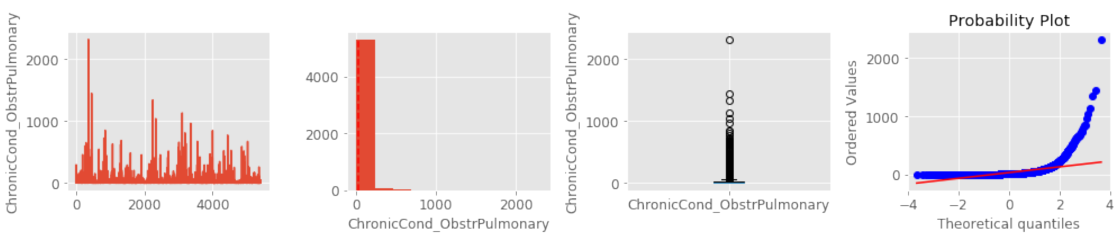
FEATURE: ChronicCond_KidneyDisease



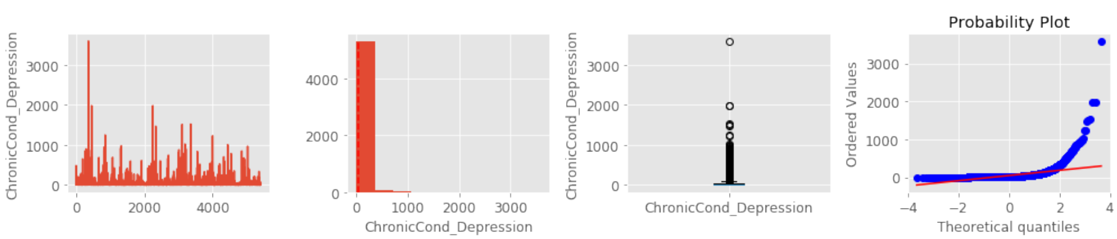
FEATURE: ChronicCond_Cancer



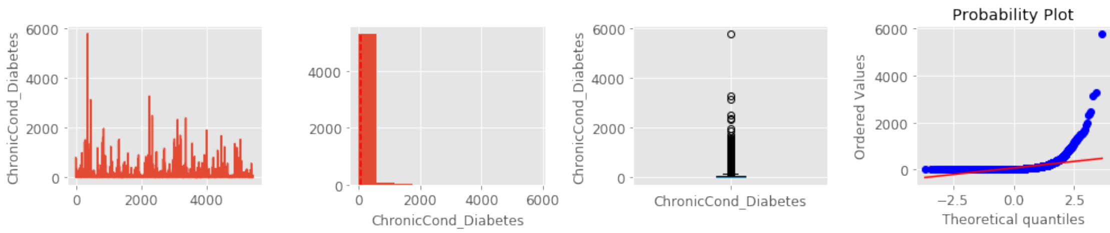
FEATURE: ChronicCond_ObstrPulmonary



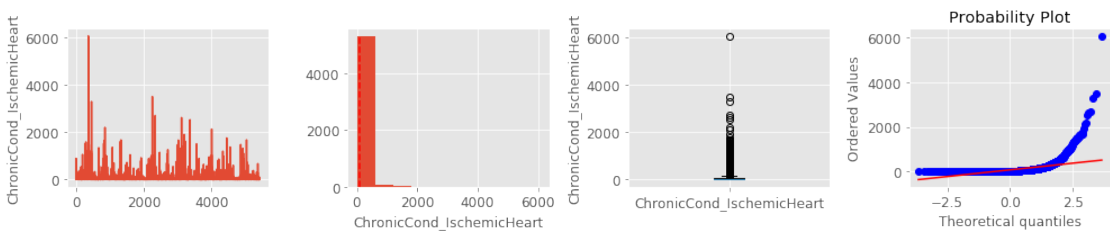
FEATURE: ChronicCond_Depression



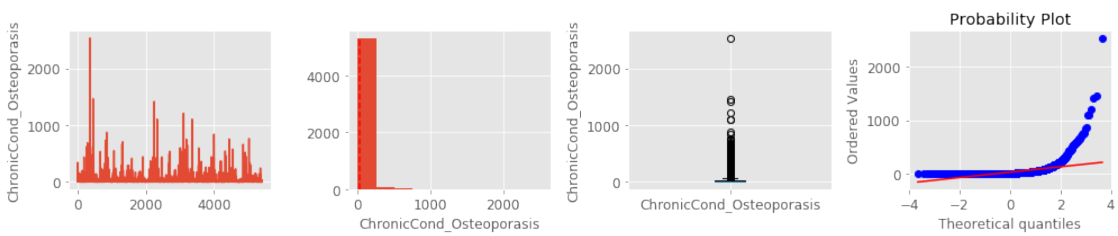
FEATURE: ChronicCond_Diabetes



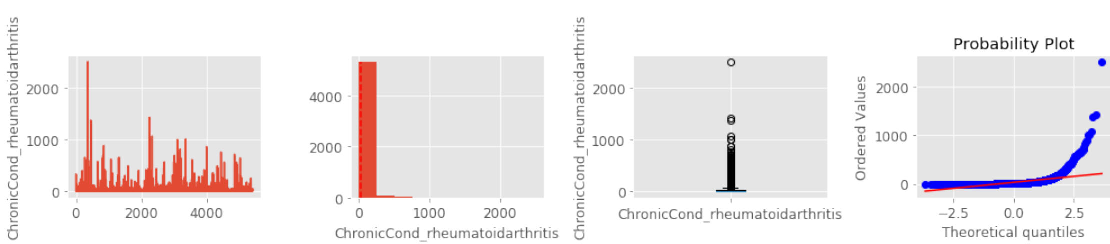
FEATURE: ChronicCond_IschemicHeart



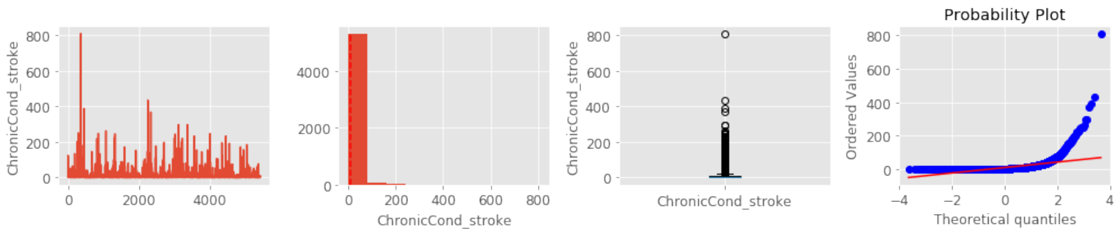
FEATURE: ChronicCond_Osteoporasis



FEATURE: ChronicCond_rheumatoidarthritis



FEATURE: ChronicCond_stroke



5 Decision Tree Classifiers

5.0.1 Model Exploration

[20]:
```
cdata.head()
```

[20]:
```
   PotentialFraud        Age  NumPhysiciansSeen  NumProcedures  \
0             0.0  78.840000           1.280000       0.120000
1             1.0  70.022727           1.181818       0.363636
2             0.0  72.161074           1.322148       0.000000
3             1.0  70.475536           1.209442       0.000000
4             0.0  69.291667           1.125000       0.013889

   NumDiagnosisClaims  InscClaimAmtReimbursed  DeductibleAmtPaid  \
0            3.640000                3.044662           1.526659
1            5.765152                3.364909           3.631355
2            2.751678               -0.004663          -0.016084
3            2.786266               -0.091330          -0.008094
4            3.208333                0.089198           0.299386

   NoOfMonths_PartACov  NoOfMonths_PartBCov  IPAnnualReimbursementAmt  …  \
0            12.000000            12.000000                  2.906059  …
1            11.818182            11.871212                  0.640754  …
2            11.865772            11.959732                 -0.085092  …
3            11.907296            11.939914                 -0.249359  …
4            11.833333            11.833333                 -0.378896  …

   ChronicCond_Heartfailure  ChronicCond_KidneyDisease  ChronicCond_Cancer  \
0                      19.0                       17.0                 5.0
1                      80.0                       64.0                10.0
2                      88.0                       50.0                16.0
3                     680.0                      507.0               165.0
4                      40.0                       22.0                12.0

   ChronicCond_ObstrPulmonary  ChronicCond_Depression  ChronicCond_Diabetes  \
```

```
0                         10.0                       9.0                    21.0
1                         41.0                      54.0                   100.0
2                         41.0                      63.0                   105.0
3                        295.0                     485.0                   799.0
4                         16.0                      29.0                    49.0

   ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                       23.0                       6.0
1                      112.0                      33.0
2                      108.0                      49.0
3                      895.0                     344.0
4                       51.0                      21.0

   ChronicCond_rheumatoidarthritis  ChronicCond_stroke
0                              8.0                 6.0
1                             38.0                12.0
2                             46.0                17.0
3                            331.0               124.0
4                             22.0                12.0

[5 rows x 24 columns]
```

[14]:
```
""" TODO
Split data into X (the inputs) and y (the outputs)

Hold out a subset of the data, before training and cross validation
using train_test_split, with stratify NOT equal to None, and a test_size
fraction of .2.

For this exploratory section, the held out set of data is a validation set.
For the GridSearch section, the held out set of data is a test set.
"""
targetnames = ['NonFraud', 'Fraud']

# TODO: Separate the data into X and y
X= cdata.drop(['PotentialFraud'], axis=1)
y= cdata['PotentialFraud']

# TODO: Split data into train and test sets
Xtrain, Xtest, ytrain, ytest= train_test_split(X, y, test_size=.2, stratify=y)
```

[15]:
```
""" TODO
Play around with the hyper-parameters. Pick your favorite model to leave with␣
↪in
your submitted report.
"""
# TODO: Create and fit the model
```

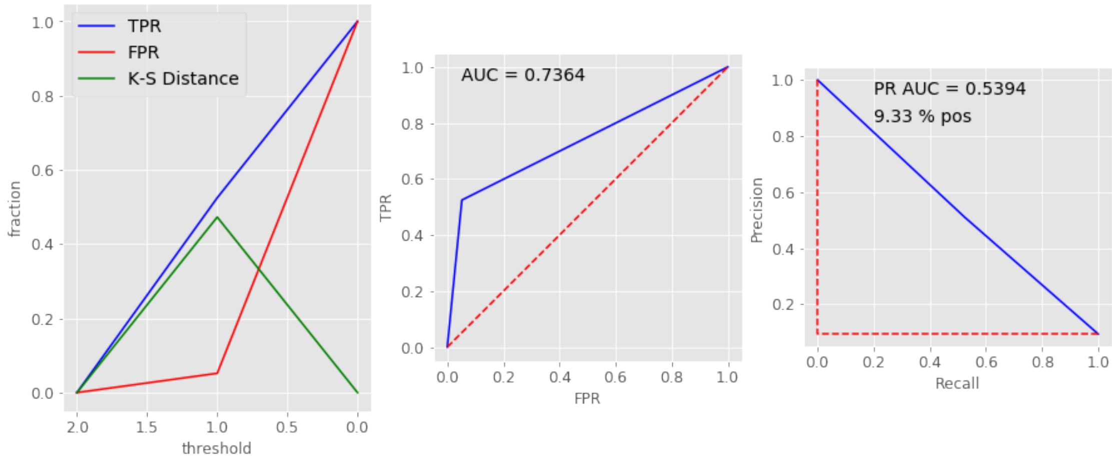
```python
model= DecisionTreeClassifier()

# TODO: Predict with the model on the validation set
model.fit(Xtrain, ytrain)
ypreds= model.predict(Xtest)

# TODO: Obtain prediction probabilities for the validation set, using
# cross_val_predict with cv=10 and method='predict_proba'
proba = cross_val_predict(model, Xtest, ytest, cv=10, method='predict_proba')


# TODO: The mean CV accuracy on the given validation data and labels, using
# cross_val_score and cv=10
scorescv = cross_val_score(model, Xtest, ytest, cv=10)# TODO
np.mean(scorescv)
```

[15]: 0.8955387205387204

[16]:
```python
""" TODO
Display the confusion matrix, KS plot, ROC curve, and PR curve for the␣
↪validation set
using metrics_plots.ks_roc_prc_plot

The red dashed line in the PRC is indicative of a the expected performance for␣
↪a random
classifier, which would predict predict postives at the rate of occurance␣
↪within the data set
"""
# TODO: Confusion Matrix
confusion= confusion_matrix(ytest, ypreds)


# TODO: Curves
# Note, you'll want the probability class predictions for the class label 1
# See the API page for the DecisionTreeClassifier predict_proba; proba_val[:,1]
preds_val= model.predict_proba(Xtest)
metrics_plots.ks_roc_prc_plot(ytest ,preds_val[:,1])


# Obtain the PSS and F1 Score
pss_val = metrics_plots.skillScore(ytest, ypreds)
f1_val = f1_score(ytest, ypreds)
print("PSS: %.4f" % pss_val[0])
print("F1 Score %.4f" % f1_val)
```

```
ROC AUC: 0.7363823538317135
```

```
PRC AUC: 0.5393650759573326
PSS: 0.4728
F1 Score 0.5171
```



[17]:
```
""" TODO
Export the image of the tree model
use export_graphviz
"""
export_graphviz(model, out_file='model.dot', rounded= True, filled= True)
```

6 GRID SEARCH CV

[18]:
```
""" TODO
Estimated time: <10 min on mlserver
Set up and run the grid search using GridSearchCV and the following
settings:
* The below scoring dictionary for scoring,
* refit set to 'f1' as the optimized metric
* Twenty for the number of cv folds,
* n_jobs=3,
* verbose=2,
* return_train_score=True
"""
# Optimized metric
opt_metric = 'f1'
scoring = {opt_metric:opt_metric}

# Flag to re-load previous run regardless of whether the file exists
force = True
```

```python
# File previous run is saved to
srchfname = "hw9_search_2" + opt_metric + ".pkl"

# SETUP EXPERIMENT HYPERPARAMETERS
max_depths = [None, 200, 100, 10, 8, 6, 4]
max_leaf_nodes = [None, 10, 5, 2]

ndepths = len(max_depths)
nleaves = len(max_leaf_nodes)

# TODO: Create the dictionary of hyper-parameters to try
hyperparams = {'max_depth': max_depths,
               'max_leaf_nodes': max_leaf_nodes}

# RUN EXPERIMENT
time0 = timelib.time()
search = None
if force or (not os.path.exists(srchfname)):
    # TODO: Create the GridSearchCV object
    dtClassifier= DecisionTreeClassifier()
    search = GridSearchCV(dtClassifier, hyperparams,␣
 ↪scoring=scoring,refit=opt_metric, cv=20, n_jobs=3, verbose=2,␣
 ↪return_train_score=True)

    # TODO: Execute the grid search by calling fit using the training data
    search.fit(Xtrain, ytrain)

    # TODO: Save the grid search object
    joblib.dump(search, srchfname)
    print("Saved %s" % srchfname)
else:
    # TODO: Re-load the grid search object
    search = joblib.load(srchfname)
    print("Loaded %s" % srchfname)

time1 = timelib.time()
duration = time1 - time0
print("Elapsed Time: %.2f min" % (duration / 60))

search
```

```
Fitting 20 folds for each of 28 candidates, totalling 560 fits
```

```
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent workers.
[Parallel(n_jobs=3)]: Done  39 tasks      | elapsed:    3.0s
[Parallel(n_jobs=3)]: Done 519 tasks      | elapsed:   12.9s
```

```
Saved hw9_search_2f1.pkl
```

```
Elapsed Time: 0.23 min

[Parallel(n_jobs=3)]: Done 555 out of 560 | elapsed:   13.5s remaining:    0.1s
[Parallel(n_jobs=3)]: Done 560 out of 560 | elapsed:   13.5s finished
```

[18]:
```
GridSearchCV(cv=20, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=3,
       param_grid={'max_depth': [None, 200, 100, 10, 8, 6, 4], 'max_leaf_nodes':
[None, 10, 5, 2]},
       pre_dispatch='2*n_jobs', refit='f1', return_train_score=True,
       scoring={'f1': 'f1'}, verbose=2)
```

7 RESULTS

[19]:
```
""" PROVIDED
Display the head of the results for the grid search
See the cv_results_ attribute
"""
all_results = search.cv_results_
df_res = pd.DataFrame(all_results)
df_res.head(3)
```

[19]:
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.106713      0.020860         0.003839        0.003879
1       0.046183      0.012744         0.002890        0.002384
2       0.035662      0.011830         0.002480        0.001819

  param_max_depth param_max_leaf_nodes  \
0            None                 None
1            None                   10
2            None                    5

                                         params  split0_test_f1  \
0  {'max_depth': None, 'max_leaf_nodes': None}        0.428571
1    {'max_depth': None, 'max_leaf_nodes': 10}        0.387097
2     {'max_depth': None, 'max_leaf_nodes': 5}        0.285714

   split1_test_f1  split2_test_f1  …  split12_train_f1  split13_train_f1  \
0        0.487805        0.545455  …          1.000000          1.000000
```

```
1        0.529412        0.526316  …         0.587065          0.593509
2        0.564103        0.444444  …         0.500888          0.517647

   split14_train_f1  split15_train_f1  split16_train_f1  split17_train_f1  \
0          1.000000          1.000000          1.000000          1.000000
1          0.612666          0.495274          0.551959          0.635854
2          0.457565          0.553459          0.450652          0.466418

   split18_train_f1  split19_train_f1  mean_train_f1  std_train_f1
0          1.000000          1.000000       1.000000      0.000000
1          0.596651          0.503759       0.570919      0.046136
2          0.493007          0.530945       0.492009      0.042948

[3 rows x 52 columns]
```

[20]:
```python
""" TODO
Obtain the best model from the grid search and
fit it to the full training data
"""
optModel= search.best_estimator_
optModel.fit(Xtrain, ytrain)
```

[20]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

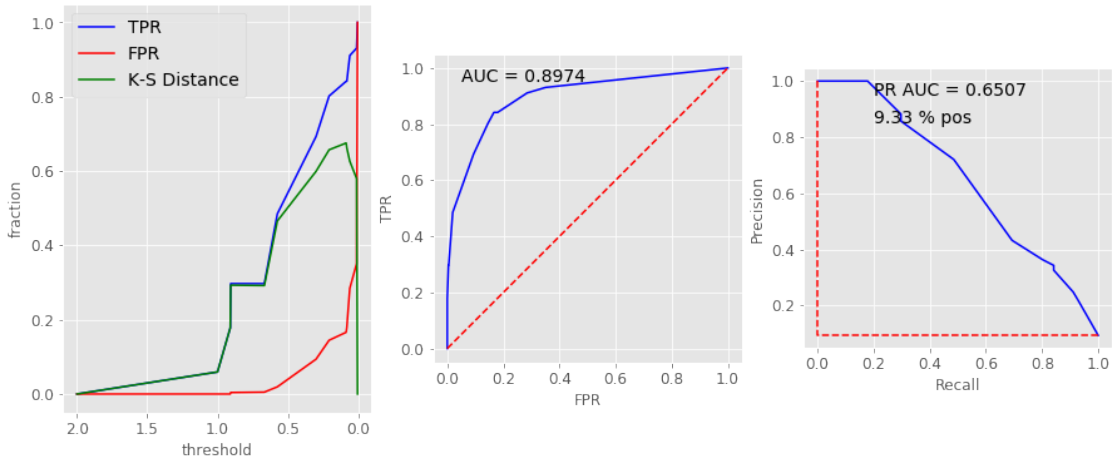
[21]:
```python
""" TODO
Export the image of the best model
use export_graphviz
"""
export_graphviz(optModel, out_file='optModel2.dot', rounded= True, filled= True)
```

[22]:
```python
""" TODO
Display the confusion matrix, KS plot, ROC curve, and PR curve for the test
set using metrics_plots.ks_roc_prc_plot

The red dashed line in the PRC is indicative of a the expected performance for
a random classifier, which would predict predict postives at the rate of
occurance within the data set
"""
# TODO: Predict with the best model on the test set
optModel.fit(Xtrain, ytrain)
ypreds= optModel.predict(Xtest)
```

```python
# TODO: Obtain prediction probabilities for the test set using cross_val_predict
# 'predict_proba' as the method
proba= cross_val_predict(optModel, Xtest, ytest, method= 'predict_proba')

# TODO: Compute mean accuracy (using cross_val_score) on the given test data␣
↪and labels
mean_acc= np.mean(cross_val_score(optModel, Xtest, ytest, cv=10))
print('mean accuracy: %.2f'%mean_acc)

# TODO: Confusion Matrix
confusion= confusion_matrix(ytest, ypreds)


# TODO: Curves (i.e. ROC, PRC, etc) use metrics_plots.ks_roc_prc_plot and the
# the probabilities for the class label of 1
proba_test= optModel.predict_proba(Xtest)
metrics_plots.ks_roc_prc_plot(ytest ,proba_test[:,1])


# Obtain the PSS and F1 Score
pss_test = metrics_plots.skillScore(ytest, ypreds)
f1_test = f1_score(ytest, ypreds)
print("PSS: %.4f" % pss_test[0])
print("F1 Score %.4f" % f1_test)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_split.py:2053:
FutureWarning: You should specify a value for 'cv' instead of relying on the
default value. The default value will change from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
```

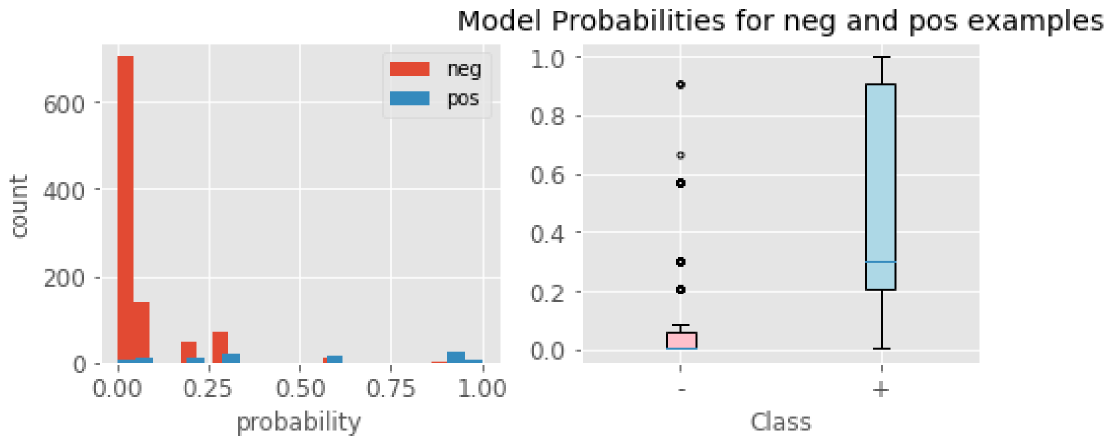
```
mean accuracy: 0.92
ROC AUC: 0.8973819400288653
PRC AUC: 0.650695364256449
PSS: 0.4658
F1 Score 0.5799
```



[23]:
```python
""" PROVIDED
Plot a histogram of the test scores from the best model.
Compare the distribution of scores for positive and negative examples
using boxplots.

Create one subplot of the distribution of all the scores, with a histogram.
Create a second subplot comparing the distribution of the scores of the
positive examples with the distribution of the negative examples, with boxplots.
"""
# Obtain the pos and neg indices
pos_inds = np.where(ytest)[0]
neg_inds = np.where(ytest == 0)[0]

# Separate the scores for the pos and neg examples
proba_pos = proba_test[pos_inds, 1]
proba_neg = proba_test[neg_inds, 1]

# Plot the distribution of all scores
nbins = 21
plt.figure(figsize=(8,3))
plt.subplot(1,2,1)
plt.hist(proba_neg, bins=nbins)
plt.hist(proba_pos, bins=nbins)
plt.xlabel('probability', fontsize=FONTSIZE)
plt.ylabel('count', fontsize=FONTSIZE)
plt.legend(['neg', 'pos'])

# Plot the boxplots of the pos and neg examples
plt.subplot(1,2,2)
boxplot = plt.boxplot([proba_neg, proba_pos], patch_artist=True, sym='.')
boxplot['boxes'][0].set_facecolor('pink')
```

```
boxplot['boxes'][1].set_facecolor('lightblue')
plt.xticks(ticks=[1, 2], labels=['-', '+'])
plt.xlabel("Class")
plt.title("Model Probabilities for neg and pos examples")
```

[23]: Text(0.5, 1.0, 'Model Probabilities for neg and pos examples')



8 Discussion

In 3 to 4 paragraphs, discuss and interpret the test results for the best model. Include a brief discussion of the histogram and boxplots of the scores. Compare the best model from the grid search to the one you chose in the exploration section. Additionally, embed the image of the best tree model into the notebook using:
`<center><img src="path_to_model.png"  style="width:100%;height:100%">`

From the histogram in the upper graph, we observe that it is getting high chance to detect negative examples and for positive examples, the results are quite spanned (slightly skewed towards 1). Also can be seen in the box plot, negative cases tends to be accumulated in the region of [0,0.1] but positive cases almost from [0,1]. It suggests even though the model can well predict negative cases, it still misunderstands positive cases in which we want more accuracy.

By neglacting the effect of unbalanced samples, meaning more negatives and less positives, we can look at the PR AUC score because it only computes at positive cases. It just reaches 0.65 for the test sample after grid search. 0.1 improvement comparing to the previous model. Thus, by calibrating these hyperparameters, our model tends to be more robust than ever.

For the exploration part, I left the default settings for the decision tree model which evolves more branches and leaves than grid search one. That is mainly the reason why it behaves worse than grid search - it overfits the data. After put constraints on maximum depths, leaf nodes, it is getting more efficient to utilize the data, and produce more robust results.