

Tree Improvement Classification models for New York City

Introduction ¶

New York city has over 680,000 trees planted on "the street" (sidewalk, medians, etc.). It's no surprise that it takes a substantial Parks Department to manage these trees, and ensure optimal health and growth. Maintaining an urban canopy is crucial to the health and success of a city's environment and human population - it's estimated that \$60M is diverted from the healthcare system annually through the existence of our urban canopy (1).

Using data from NYC Opendata, create a machine-learning model to predict whether a street tree is in need of care and/or replacement. This will provide NYC Parks department with an optimized model to prioritize trees needing care, minimizing resources to fix and increase the canopy across the city.

Data Source: <https://data.cityofnewyork.us/Environment/2015-Street-Tree-Census-Tree-Data/pi5s-9p35> (<https://data.cityofnewyork.us/Environment/2015-Street-Tree-Census-Tree-Data/pi5s-9p35>).

Outline

1. Import necessary packages, load dataset into a Pandas DataFrame, perform initial EDA
2. Creating Model's and pre-processing functions
3. Evaluation of best baseline model, fine-tuning hyperparameters
4. Final model selection, perform analysis on test data
5. Conclusion

```
In [1]: #standard packages to import
import pandas as pd
import os
import numpy as np
import scipy.stats as stats
#import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#sklearn modules
from sklearn.metrics import log_loss, confusion_matrix, accuracy_score, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from imblearn.over_sampling import SMOTE
from sklearn.utils import resample
```

```
In [2]: #nycdf = pd.read_csv('data/pluto.csv')
In [3]: treedf = pd.read_csv('data/2015StreetTreesCensus_TREES.csv')
```

```
Out[3]:
```

	created_at	tree_id	block_id	the_geom	tree_dbh	stump_diam	curb_loc	status	health
0	08/27/2015	180683	348711	POINT (-73.84421521958048 40.723091773924274)	3	0	OnCurb	Alive	F
1	09/03/2015	200540	315986	POINT (-73.81867945834878 40.79411066708779)	21	0	OnCurb	Alive	F
2	09/05/2015	204026	218365	POINT (-73.93660770459083 40.717580740099116)	3	0	OnCurb	Alive	Go

```
In [2]: #nycdf = pd.read_csv('data/pluto.csv')
In [3]: treedf.head(pd.read_csv('data/2015StreetTreesCensus_TREES.csv'))
```

Out[3]:

	created_at	tree_id	block_id	the_geom	tree_dbh	stump_diam	curb_loc	status	heal
0	08/27/2015	180683	348711	POINT (-73.84421521958048 40.723091773924274)	3	0	OnCurb	Alive	Fi
1	09/03/2015	200540	315986	POINT (-73.81867945834878 40.79411066708779)	21	0	OnCurb	Alive	Fi
2	09/05/2015	204026	218365	POINT (-73.93660770459083 40.717580740099116)	3	0	OnCurb	Alive	Go
3	09/05/2015	204337	217969	POINT (-73.93445615919741 40.713537494833226)	10	0	OnCurb	Alive	Go
4	08/30/2015	189565	223043	POINT (-73.97597938483258 40.66677775537875)	21	0	OnCurb	Alive	Go

5 rows × 42 columns

```
In [4]: #other columns to be included in model:
treedf.info()
```

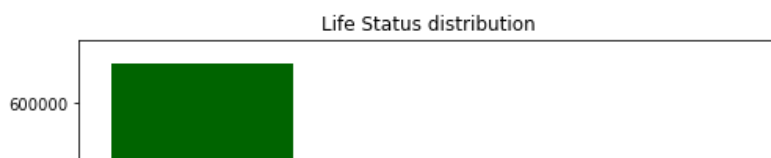
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 42 columns):
#   Column      Non-Null Count  Dtype
---  -
0   created_at  683788 non-null object
1   tree_id     683788 non-null int64
2   block_id   683788 non-null int64
3   the_geom    683788 non-null object
4   tree_dbh    683788 non-null int64
5   stump_diam  683788 non-null int64
6   curb_loc    683788 non-null object
```

```
In [4]: #other columns to be included in model:
treedf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 42 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   created_at      683788 non-null object
 1   tree_id         683788 non-null int64
 2   block_id        683788 non-null int64
 3   the_geom        683788 non-null object
 4   tree_dbh        683788 non-null int64
 5   stump_diam      683788 non-null int64
 6   curb_loc        683788 non-null object
 7   status          683788 non-null object
 8   health          652172 non-null object
 9   spc_latin       652169 non-null object
10   spc_common      652169 non-null object
11   steward         652173 non-null object
12   guards          652172 non-null object
13   sidewalk        652172 non-null object
14   user_type       683788 non-null object
15   problems        652124 non-null object
16   root_stone      683788 non-null object
17   root_grate      683788 non-null object
18   root_other      683788 non-null object
19   trnk_wire       683788 non-null object
20   trnk_light      683788 non-null object
21   trnk_other      683788 non-null object
22   brnch_ligh      683788 non-null object
23   brnch_shoe      683788 non-null object
24   brnch_othe      683788 non-null object
25   address         683788 non-null object
26   zipcode         683788 non-null int64
27   zip_city        683788 non-null object
28   cb_num          683788 non-null int64
29   borocode        683788 non-null int64
30   boroname        683788 non-null object
31   cncldist        683788 non-null int64
32   st_assem        683788 non-null int64
33   st_senate       683788 non-null int64
34   nta             683788 non-null object
35   nta_name        683788 non-null object
36   boro_ct         683788 non-null int64
37   state           683788 non-null object
38   Latitude        683788 non-null float64
39   longitude       683788 non-null float64
40   x_sp            683788 non-null float64
41   y_sp            683788 non-null float64
dtypes: float64(4), int64(11), object(27)
memory usage: 219.1+ MB
```

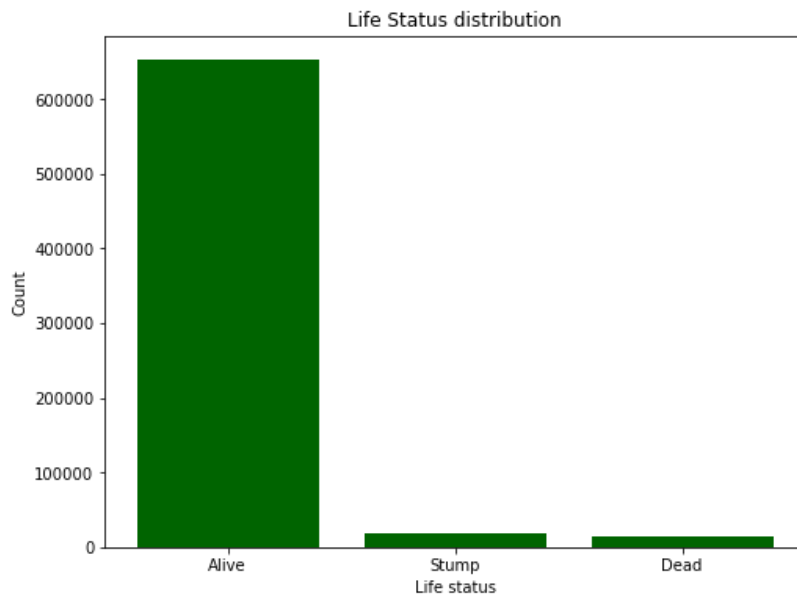
```
In [5]: bardata = treedf['status'].value_counts()

plt.figure(figsize=(8, 6))
plt.bar(bardata.index, bardata.values, color='darkgreen')
plt.xlabel('Life status')
plt.ylabel('Count')
plt.title('Life Status distribution')
plt.show()
```



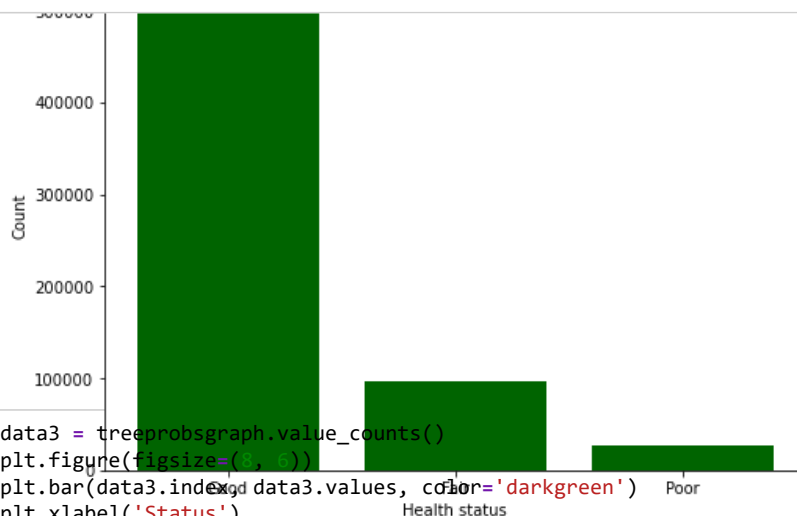
```
In [5]: bardata = treedf['status'].value_counts()

plt.figure(figsize=(8, 6))
plt.bar(bardata.index, bardata.values, color='darkgreen')
plt.xlabel('Life status')
plt.ylabel('Count')
plt.title('Life Status distribution')
plt.show()
```



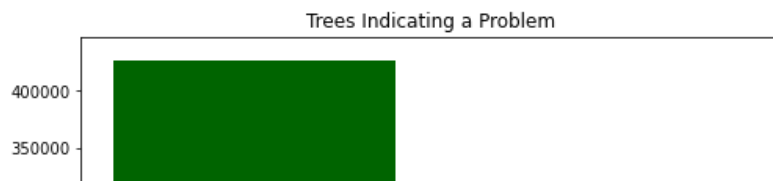
```
In [6]: bardata2 = treedf['health'].value_counts()

plt.figure(figsize=(8, 6))
plt.bar(bardata2.index, bardata2.values, color='darkgreen')
plt.xlabel('Health status')
plt.ylabel('Count')
plt.title('Health Status distribution')
plt.show()
```



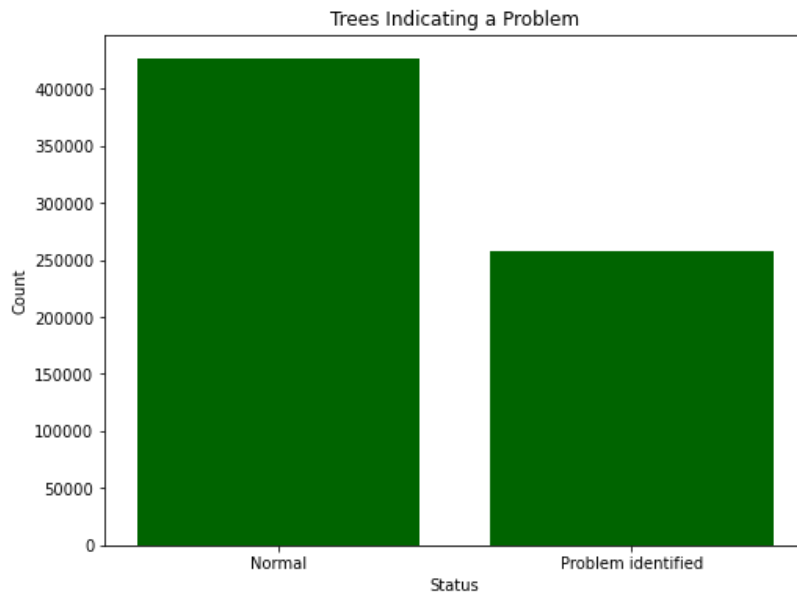
```
In [8]: data3 = treeprobsgraph.value_counts()
plt.figure(figsize=(8, 6))
plt.bar(data3.index, data3.values, color='darkgreen')
plt.xlabel('Status')
plt.ylabel('Count')
plt.title('Trees Indicating a Problem')

In [7]: treeprobsgraph = treedf['Problems'].apply(lambda x: 'Normal' if x is None or x == 'No')
plt.show()
```



```
In [8]: data3 = treeprobsgraph.value_counts()
plt.figure(figsize=(8, 6))
plt.bar(data3.index, data3.values, color='darkgreen')
plt.xlabel('Status')
plt.ylabel('Count')

In [7]: plt.title('Trees Indicating a Problem')
treeprobsgraph = treedf['problems'].apply(lambda x: 'Normal' if x is None or x == 'Problems')
plt.show()
```



Processing the main dataframe for models

```
In [9]: treedf['status'].value_counts()

Out[9]: Alive      652173
        Stump      17654
        Dead       13961
        Name: status, dtype: int64

In [10]: treedf['problems'].value_counts()

Out[10]: None                426280
         Stones              95673
         BranchLights        29452
         Stones,BranchLights  17808
         RootOther           11418
         ...
         TrunkLights,TrunkOther,BranchOther      1
         Stones,MetalGrates,RootOther,WiresRope,TrunkOther,BranchLights      1
         MetalGrates,TrunkOther,BranchLights,BranchOther      1
         MetalGrates,RootOther,TrunkLights,TrunkOther      1
         Stones,MetalGrates,RootOther,WiresRope,BranchLights      1
         Name: problems, length: 232, dtype: int64

In [13]: treedf['target_p'].value_counts()

Out[13]: 0  426280
         1  257508
         Name: target_p, dtype: int64
#creating a "target" column, 1 is tree is good, 0 is tree needs to be replaced
treedf['target'] = treedf['status'].replace({'Alive': 1, 'Stump': 0, 'Dead': 0})
#target column whether there is a problem with a tree:
treedf['target_p'] = treedf['problems'].apply(lambda x: 0 if x is None or x == 'Problems')
#manual imputing of the data - no need to keep it in original values:
treedf['root_stone'] = treedf['root_stone'].replace({'Yes': 1, 'No': 0})
treedf['root_grate'] = treedf['root_grate'].replace({'Yes': 1, 'No': 0})
treedf['health'] = treedf['health'].replace({'Good': 3, 'Fair': 2, 'Poor': 1})

Out[12]: 1      652173
         0      31615
         Name: target, dtype: int64
#brief EDA on trees where target value is "dead/stump"
dead_bar = treedf[treedf['target']==0].groupby('boroname').value_counts()
```

```

In [13]: Name: problems, length: 232, dtype: int64
treedf['target_p'] = treedf['target_p'].value_counts()

Out[13]: #creating a "target" column, 1 is tree is good, 0 is tree needs to be replaced
treedf['target'] = treedf['status'].replace({'Alive': 1, 'Stump': 0, 'Dead': 0})
Name: target_p, dtype: int64
#target column whether there is a problem with a tree:
In [14]: treedf['target_p'] = treedf['problems'].apply(lambda x: 0 if x is None or x == 'No' else 1)
#manual imputing of the data - no need to keep it in original values:
treedf['root_stone'] = treedf['root_stone'].replace({'Yes': 1, 'No': 0})
In [12]: treedf['root_grate'] = treedf['root_grate'].replace({'Yes': 1, 'No': 0})
treedf['health'] = treedf['health'].replace({'Good': 3, 'Fair': 2, 'Poor': 1})

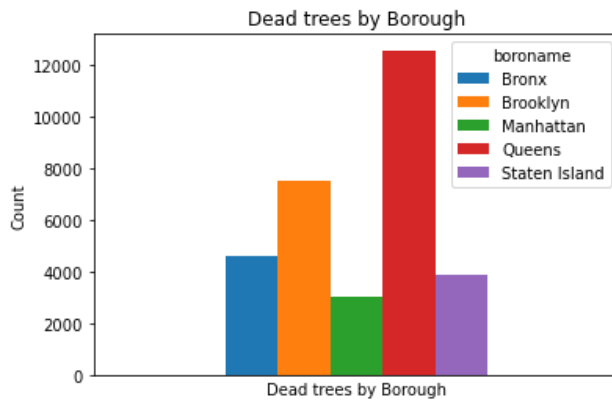
Out[12]: 1    652173
0    31615

In [15]: #brief EDA on trees where target value is "dead/stump"
dead_bar = treedf[treedf['target']==0].groupby('boroname')['boroname'].value_counts()

dead_bar.unstack().plot(kind='bar')
plt.xlabel('Dead trees by Borough')
plt.ylabel('Count')
plt.xticks([])
plt.title('Dead trees by Borough')

# Show the plot
plt.show()

```



Building the first model

```

In [16]: #Writing functions to process the data - this is fairly standard across model types

def train_process(trainset, categoricalx, numericx, classifiersx, ohex, ssx):

    #creating dummies
    train_dummies = ohex.fit_transform(trainset[categoricalx])
    # Creating the new Dataframe from OneHotEncoder
    X_train_onehot = pd.DataFrame(train_dummies, columns=ohex.get_feature_names_out())
    # Apply StandardScaler to the specified numeric columns
    trainset[numericx] = ssx.fit_transform(trainset[numericx])
    #concatenate the processed datasets
    X_train_df = pd.concat([trainset[numericx], X_train_onehot, trainset[classifiersx]]

```

Building the first model

In [16]: *#Writing functions to process the data - this is fairly standard across model types*

```
def train_process(trainset, categoricalx, numericx, classifiersx, ohex, ssx):

    #creating dummies
    train_dummies = ohex.fit_transform(trainset[categoricalx])
    # Creating the new DataFrame from OneHotEncoder
    X_train_onehot = pd.DataFrame(train_dummies, columns=ohex.get_feature_names_out())
    # Apply StandardScaler to the specified numeric columns
    trainset[numericx] = ssx.fit_transform(trainset[numericx])
    #concatenate the processed datasets
    X_train_df = pd.concat([trainset[numericx], X_train_onehot, trainset[classifiersx]])

    return X_train_df

def test_process(testset, categoricalx, numericx, classifiersx, ohex, ssx):
    #creating dummies
    test_dummies = ohex.transform(testset[categoricalx])
    # Creating the new DataFrame from OneHotEncoder
    X_test_onehot = pd.DataFrame(test_dummies, columns=ohex.get_feature_names_out())
    # Apply StandardScaler to the specified numeric columns
    testset[numericx] = ssx.transform(testset[numericx])
    #concatenate the processed datasets
    X_test_df = pd.concat([testset[numericx], X_test_onehot, testset[classifiersx]])

    return X_test_df
```

In [17]: *#deciding which columns to keep*

```
log_df = treedf.copy()

categorical = ['guards', 'steward', 'boroname', 'spc_common'] #'cb_num', 'block_id'
numeric = ['tree_dbh', 'Latitude', 'longitude']
classifiers = ['root_stone', 'root_grate', 'health', 'target_p']
target = ['target']
columns = categorical + numeric + classifiers + target

#create a subset DataFrame
log_df = log_df[columns]

#quick filter on blocks with more than 20 trees - did not end up using this, but
#block_id_counts = log_df3['block_id'].value_counts()
#log_df = log_df[log_df['block_id'].isin(block_id_counts.index[block_id_counts >= 20])]
```

```
In [17]: #deciding which columns to keep
log_df = treedf.copy()

categorical = ['guards', 'steward', 'boroname', 'spc_common'] # 'cb_num', 'block_id'
numeric = ['tree_dbh', 'Latitude', 'longitude']
classifiers = ['root_stone', 'root_grate', 'health', 'target_p']
target = ['target']
columns = categorical + numeric + classifiers + target

#create a subset DataFrame
log_df = log_df[columns]

#quick filter on blocks with more than 20 trees - did not end up using this, but
#block_id_counts = log_df3['block_id'].value_counts()
#log_df = log_df[log_df['block_id'].isin(block_id_counts.index[block_id_counts >= 20])]

#drop missing values
log_df['steward'] = log_df['steward'].fillna('No_Steward')
log_df['spc_common'] = log_df['spc_common'].fillna('Not_Avail')
#log_df['problems'] = log_df['problems'].fillna('No_problems')
log_df['health'] = log_df['health'].fillna(0)
log_df['guards'] = log_df['guards'].fillna('Unknown')

#ensure no missing values
log_df.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 12 columns):

#	Column	Non-Null Count	Dtype
0	guards	683788 non-null	object
1	steward	683788 non-null	object
2	boroname	683788 non-null	object
3	spc_common	683788 non-null	object
4	tree_dbh	683788 non-null	int64
5	Latitude	683788 non-null	float64
6	longitude	683788 non-null	float64
7	root_stone	683788 non-null	int64
8	root_grate	683788 non-null	int64
9	health	683788 non-null	float64
10	target_p	683788 non-null	int64
11	target	683788 non-null	int64

dtypes: float64(3), int64(5), object(4)
memory usage: 62.6+ MB

```
In [18]: #define the X and y variables
X = log_df.drop(target, axis=1)
y = log_df[target]

#do a train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

#instantiate a new standard scaler and one-hot encoder
ss = StandardScaler()
ohe = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

# Convert 'None' strings to a unique Label using LabelEncoder - only needed when
#keeping for future analyses
#label_encoder = LabelEncoder()
```



```
In [18]: #define the X and y variables
X = log_df.drop(target, axis=1)
y = log_df[target]

#do a train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

#instantiate a new standard scaler and one-hot encoder
ss = StandardScaler()
ohe = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

# Convert 'None' strings to a unique Label using LabelEncoder - only needed when
#keeping for future analyses
#Label_encoder = LabelEncoder()

# Apply label encoding to each column in the DataFrame - see note above
#for column in X_train.columns:
#    if X_train[column].dtype == 'O': # Check if column contains object (string)
#        X_train[column] = label_encoder.fit_transform(X_train[column])

#running the SMOTE - given large dataset, not needed.
#smote = SMOTE(random_state=53)
#X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)
```

```
In [19]: X_train_df = train_process(X_train, categorical, numeric, classifiers, ohe, ss)
X_test_df = test_process(X_test, categorical, numeric, classifiers, ohe, ss)
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preproce
ssing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output`
in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless yo
u leave `sparse` to its default value.
warnings.warn(

```
In [20]: X_train_df
```

Out[20]:

	tree_dbh	Latitude	longitude	guards_Helpful	guards_None	guards_Unknown	guards_Un
388930	-0.606654	-0.776401	-1.880071	0.0	1.0	0.0	
668243	-0.836171	-1.334534	0.768239	0.0	1.0	0.0	
406827	0.426172	-0.849981	-0.455524	0.0	1.0	0.0	
421211	0.999965	0.545221	1.073628	0.0	1.0	0.0	
300720	-0.950930	1.704516	1.082277	0.0	1.0	0.0	
...	
216253	-0.377137	-2.184911	-2.573171	0.0	1.0	0.0	
177915	1.229482	-0.861320	-0.097740	0.0	1.0	0.0	
317861	-0.606654	1.140619	-0.175556	0.0	1.0	0.0	
559989	-0.950930	-0.752730	-0.346102	1.0	0.0	0.0	
189213	-0.721413	-1.995299	-2.470034	0.0	1.0	0.0	

```
In [21]: #create the logistic regression object
log = LogisticRegression()

# Train the logistic regression model
clf = log.fit(X_train_df, y_train)

# Predict the target class based on p > 0.5 criteria
y_pred = clf.predict(X_train_df)

# Predict the probability with the training data set
clf.predict_proba(X_train_df)

# Calculate the model fit
acc1 = clf.score(X_train_df, y_train)
recall1 = recall_score(y_train, y_pred)
```

```
In [21]: # Create the logistic regression object
log = LogisticRegression()

# Train the logistic regression model
clf = log.fit(X_train_df, y_train)

# Predict the target class based on p > 0.5 criteria
y_pred = clf.predict(X_train_df)

# Predict the probability with the training data set
clf.predict_proba(X_train_df)

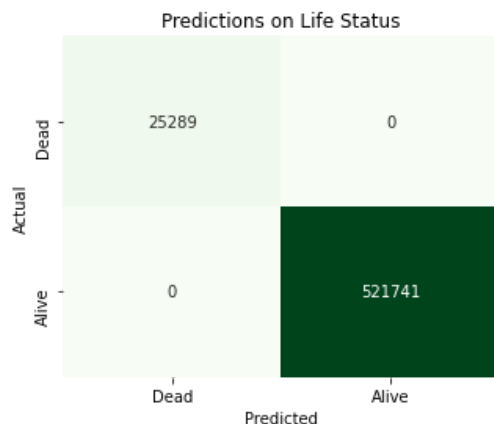
# Calculate the model fit
acc1 = clf.score(X_train_df, y_train)
recall1 = recall_score(y_train, y_pred)
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
y = column_or_1d(y, warn=True)
```

```
In [22]: # Calculate the confusion matrix
conf_matrix = confusion_matrix(y_train, y_pred)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['Dead', 'Alive'],
            yticklabels=['Dead', 'Alive'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Predictions on Life Status')
plt.show()
```



Clearly, the model has something going on.

...Turns out, almost no data is recorded for dead trees & stumps. Need to find another target variable.

Model #2 - changing the target variable

```
In [23]: #model 2 - there has to be a better way to streamline this...

categorical2 = ['steward', 'spc_common', 'status', 'cb_num', 'boroname']
numeric2 = ['tree_dbh', 'Latitude', 'longitude']
classifiers2 = ['health']
target2 = ['target_p']
columns2 = categorical2 + numeric2 + classifiers2 + target2

#create a subset DataFrame
log_df2 = treedf[columns2]

#ensure no missing values
```

Model #2 - changing the target variable

In [23]: *#model 2 - there has to be a better way to streamline this...*

```
categorical2 = ['steward', 'spc_common', 'status', 'cb_num', 'boroname']
numeric2 = ['tree_dbh', 'Latitude', 'longitude']
classifiers2 = ['health']
target2 = ['target_p']
columns2 = categorical2 + numeric2 + classifiers2 + target2

#create a subset DataFrame
log_df2 = treedf[columns2]

#ensure no missing values
log_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   steward         652173 non-null    object
1   spc_common      652169 non-null    object
2   status          683788 non-null    object
3   cb_num          683788 non-null    int64
4   boroname        683788 non-null    object
5   tree_dbh        683788 non-null    int64
6   Latitude        683788 non-null    float64
7   longitude       683788 non-null    float64
8   health          652172 non-null    float64
9   target_p        683788 non-null    int64
dtypes: float64(3), int64(3), object(4)
memory usage: 52.2+ MB
```

In [24]: *#fill na values*

```
log_df2['steward'] = log_df2['steward'].fillna('No_Steward')
log_df2['health'] = log_df2['health'].fillna(0)
log_df2['spc_common'] = log_df2['spc_common'].fillna('Not_Avail')
```

A value is trying to be set on a copy of a slice from a DataFrame.
Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df2['steward'] = log_df2['steward'].fillna('No_Steward')
<ipython-input-24-df0909064b44>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df2['health'] = log_df2['health'].fillna(0)
```

In [26]: *#python input 24-df0909064b44>:4: SettingWithCopyWarning:*

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

In [25]: *#do a train-test split*
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2, r

Out[25]: 3.0 528850
In [27]: *#instantiate a new standard scaler and one-hot encoder*

```
sc2 = StandardScaler()
one2 = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)
```

Name: health, dtype: int64
#run pre-processing functions

```
X_train_df2 = train_process(X_train2, categorical2, numeric2, classifiers2, one2,
X_test_df2 = test_process(X_test2, categorical2, numeric2, classifiers2, one2, s
```

```

log_df2['health'] = log_df2['health'].fillna(0)
In [26]: # Split the data into training and testing sets
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2, random_state=42)

In [25]: # Do a train-test split
log_df2['health'].value_counts()
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2, random_state=42)

```

```

Out[25]: 3.0    528850
          0.0    31616
          1.0    26818
In [27]: # instantiate a new standard scaler and one-hot encoder
sc2 = StandardScaler()
ohe2 = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)
Name: health, dtype: int64
# run pre-processing functions
X_train_df2 = train_process(X_train2, categorical2, numeric2, classifiers2, ohe2, sc2)
X_test_df2 = test_process(X_test2, categorical2, numeric2, classifiers2, ohe2, sc2)

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preprocessing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

```
warnings.warn(
```

```

In [28]: # Create the logistic regression object
log2 = LogisticRegression(max_iter=1000)

# Train the logistic regression model
clf2 = log2.fit(X_train_df2, y_train2)

# Predict the target class based on p > 0.5 criteria
y_pred2 = clf2.predict(X_train_df2)

# Predict the probability with the training data set
clf2.predict_proba(X_train_df2)

# Calculate the model fit
acc2 = clf2.score(X_train_df2, y_train2)

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils_validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

```
y = column_or_1d(y, warn=True)
```

```

In [29]: recall2 = recall_score(y_train2, y_pred2)
print('accuracy: '+str(acc2)+' ----- recall: '+str(recall2))

accuracy: 0.7222254720947663 ----- recall: 0.4767634552302272

```

```

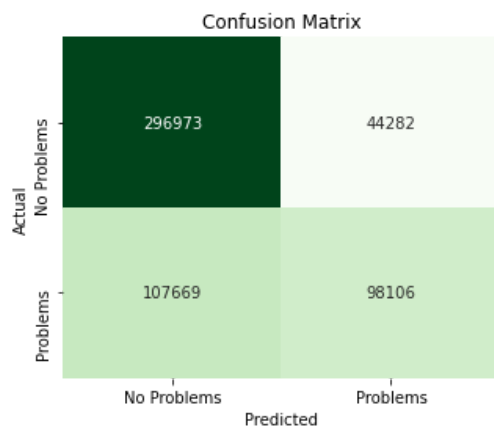
In [30]: # Calculate the confusion matrix
conf_matrix2 = confusion_matrix(y_train2, y_pred2)
labels = ('No Problems', 'Problems')
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix2, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

```

Confusion Matrix



```
In [30]: # Calculate the confusion matrix
conf_matrix2 = confusion_matrix(y_train2, y_pred2)
labels = ('No Problems', 'Problems')
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix2, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



ok, this model seems to be running average at best. Going to add more columns to see if we can reduce the false positives...

Model #3 - add other columns

```
In [31]: #model 3 - there has to be a better way to streamline this...

categorical3 = ['guards', 'steward', 'boroname', 'spc_common', 'cb_num', 'status']
numeric3 = ['tree_dbh', 'Latitude', 'longitude']
classifiers3 = ['root_stone', 'root_grate']
target3 = ['target_p']
columns3 = categorical3 + numeric3 + classifiers3 + target3

#create a subset DataFrame
log_df3 = treedf[columns3]

#ensure no missing values
```

Model #3 - add other columns

```
In [31]: #model 3 - there has to be a better way to streamline this...

categorical3 = ['guards', 'steward', 'boroname', 'spc_common', 'cb_num', 'status']
numeric3 = ['tree_dbh', 'Latitude', 'longitude']
classifiers3 = ['root_stone', 'root_grate']
target3 = ['target_p']
columns3 = categorical3 + numeric3 + classifiers3 + target3

#create a subset DataFrame
log_df3 = treedf[columns3]

#ensure no missing values
log_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 15 columns):
#   Column          Non-Null Count  Dtype
---  -
0   guards          652172 non-null  object
1   steward         652173 non-null  object
2   boroname        683788 non-null  object
3   spc_common      652169 non-null  object
4   cb_num          683788 non-null  int64
5   status          683788 non-null  object
6   sidewalk        652172 non-null  object
7   user_type       683788 non-null  object
8   cncldist        683788 non-null  int64
9   tree_dbh        683788 non-null  int64
10  Latitude         683788 non-null  float64
11  longitude        683788 non-null  float64
12  root_stone       683788 non-null  int64
13  root_grate       683788 non-null  int64
14  target_p        683788 non-null  int64
dtypes: float64(2), int64(6), object(7)
memory usage: 78.3+ MB
```

```
In [32]: #fill in null values
log_df3['guards'] = log_df3['guards'].fillna('Unknown')
log_df3['steward'] = log_df3['steward'].fillna('No_Steward')
log_df3['spc_common'] = log_df3['spc_common'].fillna('Not_Avail')
log_df3['sidewalk'] = log_df3['sidewalk'].fillna('No_issue')
```

```
<ipython-input-32-b034f7b3f9d1>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df3['guards'] = log_df3['guards'].fillna('Unknown')
```

```
In [32]: #fill in null values
log_df3['guards'] = log_df3['guards'].fillna('Unknown')
log_df3['steward'] = log_df3['steward'].fillna('No_Steward')
log_df3['spc_common'] = log_df3['spc_common'].fillna('Not_Avail')
log_df3['sidewalk'] = log_df3['sidewalk'].fillna('No_issue')
```

<ipython-input-32-b034f7b3f9d1>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df3['guards'] = log_df3['guards'].fillna('Unknown')
<ipython-input-32-b034f7b3f9d1>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df3['steward'] = log_df3['steward'].fillna('No_Steward')
<ipython-input-32-b034f7b3f9d1>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df3['spc_common'] = log_df3['spc_common'].fillna('Not_Avail')
<ipython-input-32-b034f7b3f9d1>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
log_df3['sidewalk'] = log_df3['sidewalk'].fillna('No_issue')
```

```
In [33]: log_df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 683788 entries, 0 to 683787
Data columns (total 15 columns):
#   Column          Non-Null Count  Dtype
---  -
0   guards           683788 non-null  object
1   steward          683788 non-null  object
2   boroname         683788 non-null  object
3   spc_common       683788 non-null  object
4   cb_num           683788 non-null  int64
5   status           683788 non-null  object
```

```
In [34]: #Instantiate transformers
s3 = StandardScaler()
c3 = OneHotEncoder(handle_unknown='ignore', drop = 'first', sparse = False)
tree3 = DecisionTreeClassifier()
#define the X and y variables
X1 = log_df3.drop('target', axis=1)
y3 = log_df3['target']
#do a train-test split
X_train3, X_test3, y_train3, y_test3 = train_test_split(X1, y3, test_size=0.2, random_state=42)
memory usage: 78.3+ MB
```

```
In [35]: #run the processing functions:
```

```
X_train_df3 = train_process(X_train3, categorical3, numeric3, classifiers3, ohe3, s3)
X_test_df3 = test_process(X_test3, categorical3, numeric3, classifiers3, ohe3, s3)
```

```

In [34]: #Instantiate transformers
6 sidewalk 683788 non-null object
   user_type 683788 non-null object
ss3 = StandardScaler()
8 cncidist 683788 non-null int64
ohe3 = OneHotEncoder(handle_unknown='ignore', drop = 'first', sparse = False)
9 tree_dbn 683788 non-null int64
10 Latitude 683788 non-null float64
#define the X and y variables
11 longitude 683788 non-null float64
X3 = log_df3.drop(target3, axis=1)
12 log_df3 683788 non-null int64
y3 = log_df3[target3]
13 root_grate 683788 non-null int64
14 target_p 683788 non-null int64
#do a train-test split
dtypes: float64(2), int64(6), object(7)
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=0.2, ra
memory usage: 78.3+ MB

```

```

In [35]: #run the processing functions:
X_train_df3 = train_process(X_train3, categorical3, numeric3, classifiers3, ohe3,
X_test_df3 = test_process(X_test3, categorical3, numeric3, classifiers3, ohe3, ss

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preproce
ssing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output`
in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless yo
u leave `sparse` to its default value.
warnings.warn(

```

In [36]: X_train_df3.shape

```

```

Out[36]: (547030, 263)

```

```

In [37]: # Create the logistic regression object
log3 = LogisticRegression(max_iter=10000, n_jobs=8)#, penalty='l1', solver='libl

# Train the logistic regression model
clf3 = log3.fit(X_train_df3, y_train3)

# Predict the target class based on p > 0.5 criteria
y_pred3 = clf3.predict(X_train_df3)

# Predict the probability with the training data set
clf3.predict_proba(X_train_df3)

# Calculate the model fit
acc3 = clf3.score(X_train_df3, y_train3)

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\va
lidation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,), for example
using ravel().
y = column_or_1d(y, warn=True)

```

In [38]: recall3 = recall_score(y_train3, y_pred3)
print('accuracy: '+str(acc3)+' ----- recall: '+str(recall3)+' ----- precision:

accuracy: 0.8801564813629965 ----- recall: 0.6980828574899769 ----- precisio
n: 0.976672400546645

```

```

In [39]: conf_matrix3 = confusion_matrix(y_train3, y_pred3,)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3, annot=True, cmap='Greens', cbar=False, fmt=',',
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Problems')
plt.ylabel('Actual Problems')
plt.title('Problem Classification Model')
plt.show()

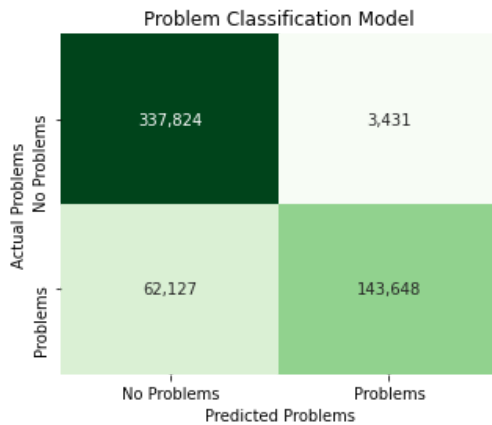
```

Problem Classification Model




```
In [39]: conf_matrix3 = confusion_matrix(y_train3, y_pred3,)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3, annot=True, cmap='Greens', cbar=False, fmt=',',
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Problems')
plt.ylabel('Actual Problems')
plt.title('Problem Classification Model')
plt.show()
```



this base-model has the highest accuracy so far, so we will tune the hyper-parameters along these columns. Using notation '--3a' for future models to prevent data-leakage.

- Next step is to try balancing the data
- Then, try tuning hyper-parameters

after running into memory issues running the model, we need to down-sample the majority class. once the model runs successfully, will tune the amount of data to increase as much as possible before running into the memory issue

```
In [40]: # Separate majority and minority classes
majority_class = log_df3[log_df3['target_p'] == 0]
minority_class = log_df3[log_df3['target_p'] == 1]

# Downsample majority class
downsampled_majority = resample(majority_class, replace=False, n_samples=len(minority_class))

# Combine the dataframes
balanced_df = pd.concat([downsampled_majority, minority_class])

# Shuffle the rows
log_df3_bal = balanced_df.sample(frac=1, random_state=53).reset_index(drop=True)
```

```
In [41]: log_df3_bal['target_p'].value_counts()
```

```
Out[41]: 0    257508
         1    62127
         2     3431
         3      143648
         4      143648
         5      143648
         6      143648
         7      143648
         8      143648
         9      143648
         10     143648
         11     143648
         12     143648
         13     143648
         14     143648
         15     143648
         16     143648
         17     143648
         18     143648
         19     143648
         20     143648
         21     143648
         22     143648
         23     143648
         24     143648
         25     143648
         26     143648
         27     143648
         28     143648
         29     143648
         30     143648
         31     143648
         32     143648
         33     143648
         34     143648
         35     143648
         36     143648
         37     143648
         38     143648
         39     143648
         40     143648
         41     143648
         42     143648
         43     143648
         44     143648
         45     143648
         46     143648
         47     143648
         48     143648
         49     143648
         50     143648
         51     143648
         52     143648
         53     143648
         54     143648
         55     143648
         56     143648
         57     143648
         58     143648
         59     143648
         60     143648
         61     143648
         62     143648
         63     143648
         64     143648
         65     143648
         66     143648
         67     143648
         68     143648
         69     143648
         70     143648
         71     143648
         72     143648
         73     143648
         74     143648
         75     143648
         76     143648
         77     143648
         78     143648
         79     143648
         80     143648
         81     143648
         82     143648
         83     143648
         84     143648
         85     143648
         86     143648
         87     143648
         88     143648
         89     143648
         90     143648
         91     143648
         92     143648
         93     143648
         94     143648
         95     143648
         96     143648
         97     143648
         98     143648
         99     143648
         100    143648
         101    143648
         102    143648
         103    143648
         104    143648
         105    143648
         106    143648
         107    143648
         108    143648
         109    143648
         110    143648
         111    143648
         112    143648
         113    143648
         114    143648
         115    143648
         116    143648
         117    143648
         118    143648
         119    143648
         120    143648
         121    143648
         122    143648
         123    143648
         124    143648
         125    143648
         126    143648
         127    143648
         128    143648
         129    143648
         130    143648
         131    143648
         132    143648
         133    143648
         134    143648
         135    143648
         136    143648
         137    143648
         138    143648
         139    143648
         140    143648
         141    143648
         142    143648
         143    143648
         144    143648
         145    143648
         146    143648
         147    143648
         148    143648
         149    143648
         150    143648
         151    143648
         152    143648
         153    143648
         154    143648
         155    143648
         156    143648
         157    143648
         158    143648
         159    143648
         160    143648
         161    143648
         162    143648
         163    143648
         164    143648
         165    143648
         166    143648
         167    143648
         168    143648
         169    143648
         170    143648
         171    143648
         172    143648
         173    143648
         174    143648
         175    143648
         176    143648
         177    143648
         178    143648
         179    143648
         180    143648
         181    143648
         182    143648
         183    143648
         184    143648
         185    143648
         186    143648
         187    143648
         188    143648
         189    143648
         190    143648
         191    143648
         192    143648
         193    143648
         194    143648
         195    143648
         196    143648
         197    143648
         198    143648
         199    143648
         200    143648
         201    143648
         202    143648
         203    143648
         204    143648
         205    143648
         206    143648
         207    143648
         208    143648
         209    143648
         210    143648
         211    143648
         212    143648
         213    143648
         214    143648
         215    143648
         216    143648
         217    143648
         218    143648
         219    143648
         220    143648
         221    143648
         222    143648
         223    143648
         224    143648
         225    143648
         226    143648
         227    143648
         228    143648
         229    143648
         230    143648
         231    143648
         232    143648
         233    143648
         234    143648
         235    143648
         236    143648
         237    143648
         238    143648
         239    143648
         240    143648
         241    143648
         242    143648
         243    143648
         244    143648
         245    143648
         246    143648
         247    143648
         248    143648
         249    143648
         250    143648
         251    143648
         252    143648
         253    143648
         254    143648
         255    143648
         256    143648
         257    143648
         258    143648
         259    143648
         260    143648
         261    143648
         262    143648
         263    143648
         264    143648
         265    143648
         266    143648
         267    143648
         268    143648
         269    143648
         270    143648
         271    143648
         272    143648
         273    143648
         274    143648
         275    143648
         276    143648
         277    143648
         278    143648
         279    143648
         280    143648
         281    143648
         282    143648
         283    143648
         284    143648
         285    143648
         286    143648
         287    143648
         288    143648
         289    143648
         290    143648
         291    143648
         292    143648
         293    143648
         294    143648
         295    143648
         296    143648
         297    143648
         298    143648
         299    143648
         300    143648
         301    143648
         302    143648
         303    143648
         304    143648
         305    143648
         306    143648
         307    143648
         308    143648
         309    143648
         310    143648
         311    143648
         312    143648
         313    143648
         314    143648
         315    143648
         316    143648
         317    143648
         318    143648
         319    143648
         320    143648
         321    143648
         322    143648
         323    143648
         324    143648
         325    143648
         326    143648
         327    143648
         328    143648
         329    143648
         330    143648
         331    143648
         332    143648
         333    143648
         334    143648
         335    143648
         336    143648
         337    143648
         338    143648
         339    143648
         340    143648
         341    143648
         342    143648
         343    143648
         344    143648
         345    143648
         346    143648
         347    143648
         348    143648
         349    143648
         350    143648
         351    143648
         352    143648
         353    143648
         354    143648
         355    143648
         356    143648
         357    143648
         358    143648
         359    143648
         360    143648
         361    143648
         362    143648
         363    143648
         364    143648
         365    143648
         366    143648
         367    143648
         368    143648
         369    143648
         370    143648
         371    143648
         372    143648
         373    143648
         374    143648
         375    143648
         376    143648
         377    143648
         378    143648
         379    143648
         380    143648
         381    143648
         382    143648
         383    143648
         384    143648
         385    143648
         386    143648
         387    143648
         388    143648
         389    143648
         390    143648
         391    143648
         392    143648
         393    143648
         394    143648
         395    143648
         396    143648
         397    143648
         398    143648
         399    143648
         400    143648
         401    143648
         402    143648
         403    143648
         404    143648
         405    143648
         406    143648
         407    143648
         408    143648
         409    143648
         410    143648
         411    143648
         412    143648
         413    143648
         414    143648
         415    143648
         416    143648
         417    143648
         418    143648
         419    143648
         420    143648
         421    143648
         422    143648
         423    143648
         424    143648
         425    143648
         426    143648
         427    143648
         428    143648
         429    143648
         430    143648
         431    143648
         432    143648
         433    143648
         434    143648
         435    143648
         436    143648
         437    143648
         438    143648
         439    143648
         440    143648
         441    143648
         442    143648
         443    143648
         444    143648
         445    143648
         446    143648
         447    143648
         448    143648
         449    143648
         450    143648
         451    143648
         452    143648
         453    143648
         454    143648
         455    143648
         456    143648
         457    143648
         458    143648
         459    143648
         460    143648
         461    143648
         462    143648
         463    143648
         464    143648
         465    143648
         466    143648
         467    143648
         468    143648
         469    143648
         470    143648
         471    143648
         472    143648
         473    143648
         474    143648
         475    143648
         476    143648
         477    143648
         478    143648
         479    143648
         480    143648
         481    143648
         482    143648
         483    143648
         484    143648
         485    143648
         486    143648
         487    143648
         488    143648
         489    143648
         490    143648
         491    143648
         492    143648
         493    143648
         494    143648
         495    143648
         496    143648
         497    143648
         498    143648
         499    143648
         500    143648
         501    143648
         502    143648
         503    143648
         504    143648
         505    143648
         506    143648
         507    143648
         508    143648
         509    143648
         510    143648
         511    143648
         512    143648
         513    143648
         514    143648
         515    143648
         516    143648
         517    143648
         518    143648
         519    143648
         520    143648
         521    143648
         522    143648
         523    143648
         524    143648
         525    143648
         526    143648
         527    143648
         528    143648
         529    143648
         530    143648
         531    143648
         532    143648
         533    143648
         534    143648
         535    143648
         536    143648
         537    143648
         538    143648
         539    143648
         540    143648
         541    143648
         542    143648
         543    143648
         544    143648
         545    143648
         546    143648
         547    143648
         548    143648
         549    143648
         550    143648
         551    143648
         552    143648
         553    143648
         554    143648
         555    143648
         556    143648
         557    143648
         558    143648
         559    143648
         560    143648
         561    143648
         562    143648
         563    143648
         564    143648
         565    143648
         566    143648
         567    143648
         568    143648
         569    143648
         570    143648
         571    143648
         572    143648
         573    143648
         574    143648
         575    143648
         576    143648
         577    143648
         578    143648
         579    143648
         580    143648
         581    143648
         582    143648
         583    143648
         584    143648
         585    143648
         586    143648
         587    143648
         588    143648
         589    143648
         590    143648
         591    143648
         592    143648
         593    143648
         594    143648
         595    143648
         596    143648
         597    143648
         598    143648
         599    143648
         600    143648
         601    143648
         602    143648
         603    143648
         604    143648
         605    143648
         606    143648
         607    143648
         608    143648
         609    143648
         610    143648
         611    143648
         612    143648
         613    143648
         614    143648
         615    143648
         616    143648
         617    143648
         618    143648
         619    143648
         620    143648
         621    143648
         622    143648
         623    143648
         624    143648
         625    143648
         626    143648
         627    143648
         628    143648
         629    143648
         630    143648
         631    143648
         632    143648
         633    143648
         634    143648
         635    143648
         636    143648
         637    143648
         638    143648
         639    143648
         640    143648
         641    143648
         642    143648
         643    143648
         644    143648
         645    143648
         646    143648
         647    143648
         648    143648
         649    143648
         650    143648
         651    143648
         652    143648
         653    143648
         654    143648
         655    143648
         656    143648
         657    143648
         658    143648
         659    143648
         660    143648
         661    143648
         662    143648
         663    143648
         664    143648
         665    143648
         666    143648
         667    143648
         668    143648
         669    143648
         670    143648
         671    143648
         672    143648
         673    143648
         674    143648
         675    143648
         676    143648
         677    143648
         678    143648
         679    143648
         680    143648
         681    143648
         682    143648
         683    143648
         684    143648
         685    143648
         686    143648
         687    143648
         688    143648
         689    143648
         690    143648
         691    143648
         692    143648
         693    143648
         694    143648
         695    143648
         696    143648
         697    143648
         698    143648
         699    143648
         700    143648
         701    143648
         702    143648
         703    143648
         704    143648
         705    143648
         706    143648
         707    143648
         708    143648
         709    143648
         710    143648
         711    143648
         712    143648
         713    143648
         714    143648
         715    143648
         716    143648
         717    143648
         718    143648
         719    143648
         720    143648
         721    143648
         722    143648
         723    143648
         724    143648
         725    143648
         726    143648
         727    143648
         728    143648
         729    143648
         730    143648
         731    143648
         732    143648
         733    143648
         734    143648
         735    143648
         736    143648
         737    143648
         738    143648
         739    143648
         740    143648
         741    143648
         742    143648
         743    143648
         744    143648
         745    143648
         746    143648
         747    143648
         748    143648
         749    143648
         750    143648
         751    143648
         752    143648
         753    143648
         754    143648
         755    143648
         756    143648
         757    143648
         758    143648
         759    143648
         760    143648
         761    143648
         762    143648
         763    143648
         764    143648
         765    143648
         766    143648
         767    143648
         768    143648
         769    143648
         770    143648
         771    143648
         772    143648
         773    143648
         774    143648
         775    143648
         776    143648
         777    143648
         778    143648
         779    143648
         780    143648
         781    143648
         782    143648
         783    143648
         784    143648
         785    143648
         786    143648
         787    143648
         788    143648
         789    143648
         790    143648
         791    143648
         792    143648
         793    143648
         794    143648
         795    143648
         796    143648
         797    143648
         798    143648
         799    143648
         800    143648
         801    143648
         802    143648
         803    143648
         804    143648
         805    143648
         806    143648
         807    143648
         808    143648
         809    143648
         810    143648
         811    143648
         812    143648
         813    143648
         814    143648
         815    143648
         816    143648
         817    143648
         818    143648
         819    143648
         820    143648
         821    143
```

```

Out[42]: #Install the transformers
os3a = StandardScaler()
ana3a = TargetEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

#define the X and y variables
X3a = log_df3_bal.drop(target3, axis=1)
y3a = log_df3_bal[target3]

#do a train-test split
X_train3a, X_test3a, y_train3a, y_test3a = train_test_split(X3a, y3a, test_size=0.2)

```

```

In [43]: #run the processing functions:
X_train_df3a = train_process(X_train3a, categorical3, numeric3, classifiers3, ohe3a)
X_test_df3a = test_process(X_test3a, categorical3, numeric3, classifiers3, ohe3a)

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preproce
ssing\_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output`
in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless yo
u leave `sparse` to its default value.
  warnings.warn(

```

```

In [44]: X_train_df3a.shape

```

```

Out[44]: (412012, 263)

```

```

In [45]: # Create the Logistic regression object
log3a = LogisticRegression(max_iter=10000, n_jobs=8)

# Train the Logistic regression model
clf3a = log3a.fit(X_train_df3a, y_train3a)

# Predict the target class based on p > 0.5 criteria
y_pred3a = clf3a.predict(X_train_df3a)

# Predict the probability with the training data set
clf3a.predict_proba(X_train_df3a)

# Calculate the model fit
acc3a = clf3a.score(X_train_df3a, y_train3a)

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\va
lidation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)

```

```

In [46]: recall3a = recall_score(y_train3a, y_pred3a)
print('accuracy: '+str(acc3a)+' ----- recall: '+str(recall3a))

accuracy: 0.8478976340494937 ----- recall: 0.7358698608844331

```

```

In [47]: conf_matrix3a = confusion_matrix(y_train3a, y_pred3a)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3a, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Health')
plt.ylabel('Actual Health')
plt.title('Problems Classification Model')
plt.show()

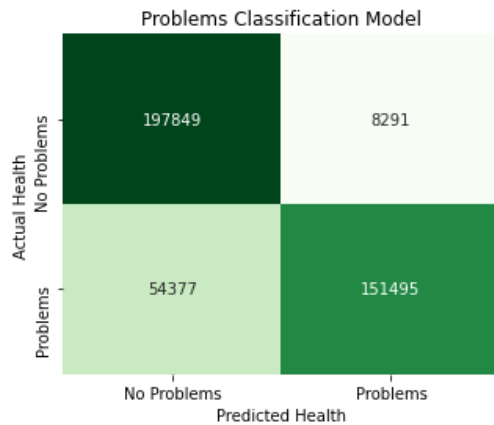
```

Problems Classification Model



```
In [47]: conf_matrix3a = confusion_matrix(y_train3a, y_pred3a)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3a, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Health')
plt.ylabel('Actual Health')
plt.title('Problems Classification Model')
plt.show()
```



Balancing the dataset did not help the model, so we will continue with the unbalanced dataset.

Tuning hyper-parameters on model #3

```
In [48]: #starting with C-value
#do a train-test split
X_train3b, X_test3b, y_train3b, y_test3b = train_test_split(X3, y3, test_size=0.2)

#set new processors
ss3b = StandardScaler()
ohe3b = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

#run the processing functions:
X_train_df3b = train_process(X_train3b, categorical3, numeric3, classifiers3, ohe3b)
X_test_df3b = test_process(X_test3b, categorical3, numeric3, classifiers3, ohe3b)
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preprocessing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

warnings.warn(

```
In [49]: X_train_df3b.shape
In [50]: # Create the logistic regression object
log3b = LogisticRegression(max_iter=10000, n_jobs=8, C=100)
Out[49]: (547030, 263)

# Train the logistic regression model
clf3b = log3b.fit(X_train_df3b, y_train3b)

# Predict the target class based on p > 0.5 criteria
y_pred3b = clf3b.predict(X_train_df3b)

# Predict the probability with the training data set
prob3b = clf3b.predict_proba(X_train_df3b)

# Calculate the model fit
acc3b = clf3b.score(X_train_df3b, y_train3b)
```

```

In [49]: X_train_df3b.shape
In [50]: # Create the Logistic regression object
log3b = LogisticRegression(max_iter=10000, n_jobs=8, C=100)
Out[49]: (547030, 263)

# Train the Logistic regression model
clf3b = log3b.fit(X_train_df3b, y_train3b)

# Predict the target class based on p > 0.5 criteria
y_pred3b = clf3b.predict(X_train_df3b)

# Predict the probability with the training data set
prob3b = clf3b.predict_proba(X_train_df3b)

# Calculate the model fit
acc3b = clf3b.score(X_train_df3b, y_train3b)

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

y = column_or_1d(y, warn=True)

```

In [51]: recall3b = recall_score(y_train3b, y_pred3b)
print('accuracy: '+str(acc3b)+' ----- recall: '+str(recall3b))

```

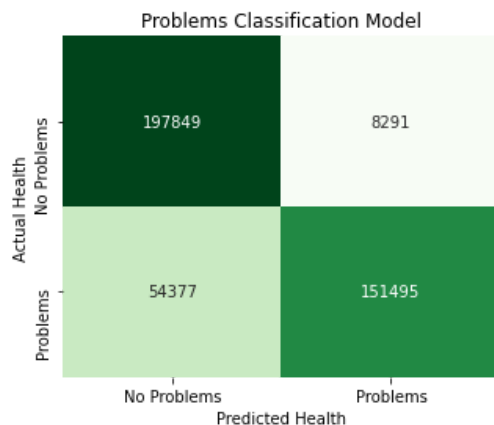
accuracy: 0.8801692777361387 ----- recall: 0.6982043494107641

```

In [52]: conf_matrix3b = confusion_matrix(y_train3b, y_pred3b)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3a, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Health')
plt.ylabel('Actual Health')
plt.title('Problems Classification Model')
plt.show()

```



```

In [53]: Changing C worked better, now shifting from lasso to ridge penalty:
#set to 100
#next hyper-parameter: change from L2 to L1 penalty
#do a train-test split
X_train3c, X_test3c, y_train3c, y_test3c = train_test_split(X3, y3, test_size=0.2)

#set new processors
ss3c = StandardScaler()
ohe3c = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

X_train_df3c = train_process(X_train3c, categorical3, numeric3, classifiers3, ohe3c)
X_test_df3c = test_process(X_test3c, categorical3, numeric3, classifiers3, ohe3c)

# Create the Logistic regression object with penalty L1
log3c = LogisticRegression(max_iter=10000, penalty='l1', solver='liblinear', C=100)

```

```

In [53]: Changing C worked better, now shifting from lasso to ridge penalty:
#set C to 100
#next hyper-parameter: change from L2 to L1 penalty
#do a train-test split
X_train3c, X_test3c, y_train3c, y_test3c = train_test_split(X3, y3, test_size=0.2)

#set new processors
ss3c = StandardScaler()
ohe3c = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

X_train_df3c = train_process(X_train3c, categorical3, numeric3, classifiers3, ohe3c)
X_test_df3c = test_process(X_test3c, categorical3, numeric3, classifiers3, ohe3c)

# Create the Logistic regression object with penalty L1
log3c = LogisticRegression(max_iter=10000, penalty='l1', solver='liblinear', C=100)

# Train the Logistic regression model
clf3c = log3c.fit(X_train_df3c, y_train3c)
# Predict the target class based on p > 0.5 criteria
y_pred3c = clf3c.predict(X_train_df3c)
# Predict the probability with the training data set
clf3c.predict_proba(X_train_df3c)
# Calculate the model fit
acc3c = clf3c.score(X_train_df3c, y_train3c)

```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preprocessing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

warnings.warn(
C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\validation.py:1183: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

```

In [54]: recall3c = recall_score(y_train3c, y_pred3c)
print('accuracy: '+str(acc3c)+' ----- recall: '+str(recall3c))

```

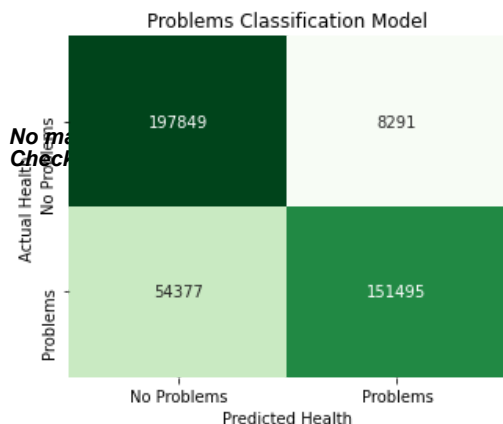
accuracy: 0.8801674496828328 ----- recall: 0.6981897703802697

```

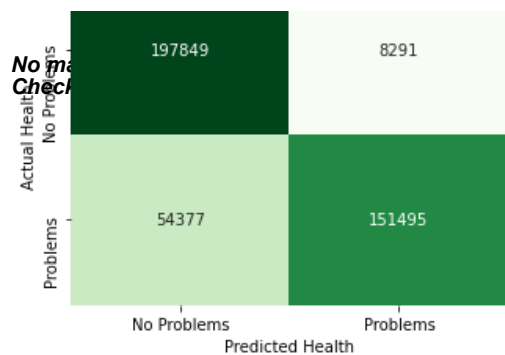
In [55]: conf_matrix3b = confusion_matrix(y_train3b, y_pred3b)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3a, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Health')
plt.ylabel('Actual Health')
plt.title('Problems Classification Model')
plt.show()

```



nalties, will stick with lasso for simplicity.



...nalties, will stick with lasso for simplicity.

```
In [56]: #check cross-validation
log3d = LogisticRegression(max_iter=10000)

#do a train-test split
X_train3d, X_test3d, y_train3d, y_test3d = train_test_split(X3, y3, test_size=0.2)

#set new processors
ss3d = StandardScaler()
ohe3d = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

#run the processing functions:
X_train_df3d = train_process(X_train3d, categorical3, numeric3, classifiers3, ohe3d)
X_test_df3d = test_process(X_test3d, categorical3, numeric3, classifiers3, ohe3d)

#5-fold cross-validation
```

```
In [56]: #check cross-validation
log3d = LogisticRegression(max_iter=10000)

#do a train-test split
X_train3d, X_test3d, y_train3d, y_test3d = train_test_split(X3, y3, test_size=0.2)

#set new processors
ss3d = StandardScaler()
ohe3d = OneHotEncoder(handle_unknown="ignore", drop = 'first', sparse = False)

#run the processing functions:
X_train_df3d = train_process(X_train3d, categorical3, numeric3, classifiers3, ohe3d)
X_test_df3d = test_process(X_test3d, categorical3, numeric3, classifiers3, ohe3d)

#5-fold cross-validation
log3d = LogisticRegression(max_iter=10000, n_jobs=8, C=100)
scores = cross_validate(log3d, X_train_df3d, y_train3d, cv=5, n_jobs=8)

# Print cross-validation scores and mean score
print("Cross-Validation Scores:", scores)
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\preprocessing_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

```
warnings.warn(
```

```
Cross-Validation Scores: {'fit_time': array([332.80013156, 371.11125803, 40.2144196, 412.00748277, 395.5266819 ]), 'score_time': array([0.62764955, 0.53343058, 0.42180777, 0.4061532 ]), 'test_score': array([0.87909255, 0.87970495, 0.87968667, 0.88103943])}
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selection_validation.py:425: FitFailedWarning:

```
1 fits failed out of a total of 5.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.
```

Below are more details about the failures:

```
-----
-
1 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
```

C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selection_validation.py:425: FitFailedWarning:
1 fits failed out of a total of 5.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:

```
-----
-
1 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\model_selection\_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\base.py", line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\linear_model\_logistic.py", line 1208, in fit
    X, y = self._validate_data(
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\base.py", line 622, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\_validation.py", line 1146, in check_X_y
    X = check_array(
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\_validation.py", line 915, in check_array
    array = _asarray_with_order(array, order=order, dtype=dtype, xp=xp)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\sklearn\utils\_array_api.py", line 380, in _asarray_with_order
    array = numpy.asarray(array, order=order, dtype=dtype)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\pandas\core\generic.py", line 1781, in __array__
    return np.asarray(self.values, dtype=dtype)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\pandas\core\generic.py", line 5348, in _values
    return self.values
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\pandas\core\generic.py", line 5343, in values
    return self._mgr.as_array(transpose=self._AXIS_REVERSED)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\pandas\core\internals\managers.py", line 853, in as_array
    arr = self._interleave(dtype=dtype, na_value=na_value)
  File "C:\Users\Reid Majka\anaconda3\envs\learn-env\lib\site-packages\pandas\core\internals\managers.py", line 882, in _interleave
    result = np.empty(self.shape, dtype=dtype)
numpy.core._exceptions._ArrayMemoryError: Unable to allocate 878. MiB for an array with shape (263, 437624) and data type float64

warnings.warn(some_fits_failed_message, FitFailedWarning)
```

Try a Decision Tree classifier on model #3

```
In [50]: tree_df = tree_df.drop(columns=target3)
X4 = tree_df.drop(target3, axis=1)
y4 = tree_df[target3]
In [58]: # Convert 'None' strings to a unique label using LabelEncoder
label_encoder = LabelEncoder()
X_train4, X_test4, y_train4, y_test4 = train_test_split(X4, y4, test_size=0.2, random_state=53)
# Apply label encoding to each column in the DataFrame
for column in X_train4.columns:
    if tree_df[column].dtype == 'O': # Check if the column contains object (string) data
        tree_clf.fit(X_train4[y_train4.index], y_train4)
```

```
Out[59]: DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=53)
```



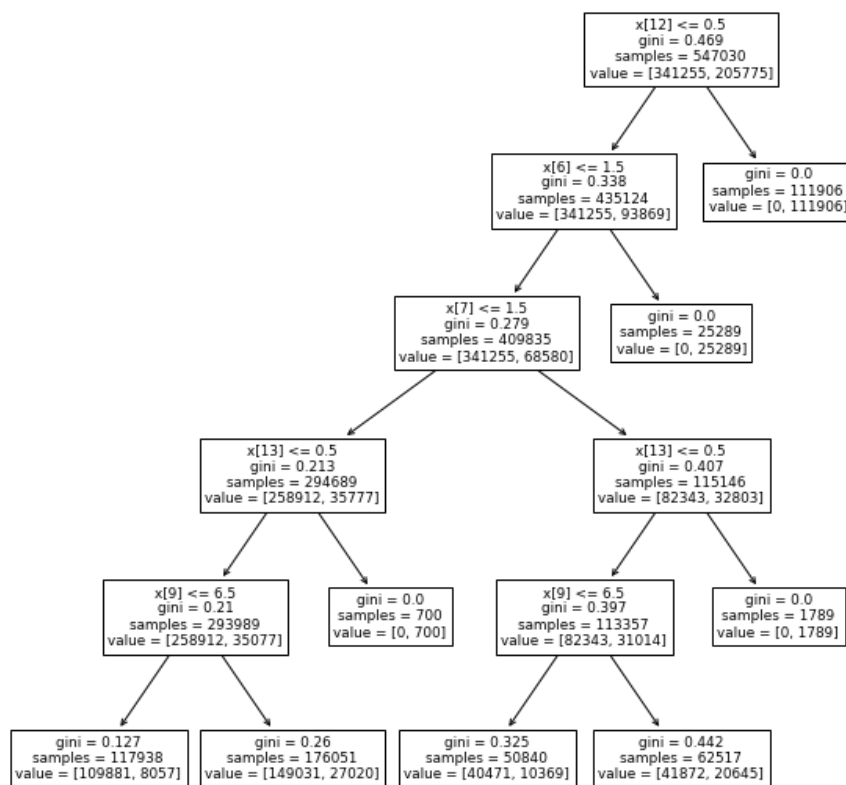
```
In [57]: tree_df = tree_df.drop(columns=[target3])
X4 = tree_df.drop(target3, axis=1)
y4 = tree_df[target3]
In [58]: # Convert 'None' strings to a unique Label using LabelEncoder
label_encoder = LabelEncoder()
X_train4, X_test4, y_train4, y_test4 = train_test_split(X4, y4, test_size=0.2, random_state=53)
# Apply Label encoding to each column in the DataFrame
for column in X_train4.columns:
    if tree_df[column].dtype == 'O': # Check if the column contains object (string)
        tree_clf.fit(X_train4[y_train4==1], y_train4[y_train4==1])
        label_encoder.fit_transform(tree_df[column])
```

```
Out[59]: DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=53)
```

```
In [60]: X_train4.shape
```

```
Out[60]: (547030, 14)
```

```
In [61]: f, ax = plt.subplots(figsize=(10, 10))
plot_tree(tree_clf, ax=ax);
```



```
In [64]: acc = accuracy_score(y_train4, y_pred4) * 100
In [62]: y_pred4 = tree_clf.predict(X_train4)
print("Accuracy score: {}".format(acc))
print("Recall score: {}".format(recall(y_train4, y_pred4)))
```

```
In [63]: Accuracy: 87.91821289508802
Recall: 0.6788161588299477
```

```
In [65]: conf_matrix4 = confusion_matrix(y_train4, y_pred4)
Out[63]: array([[0, 0, 1, ..., 0, 0, 0], dtype=int64])

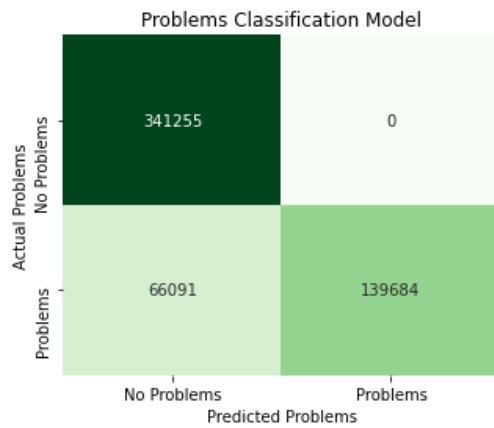
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix4, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Problems')
```

```
In [64]: acc = accuracy_score(y_train4, y_pred4) * 100
In [62]: y_pred4 = recall_score(y_train4, y_pred4)
print("Accuracy: {}%".format(acc))
print("Recall: {}%".format(recall))
```

```
In [63]: X_train4 = X_train[X_train['Problems'] == 0]
y_pred4
```

```
Out[63]: conf_matrix4 = confusion_matrix(y_train4, y_pred4)
array([[0, 0, 1, ..., 0, 0, 0], dtype=int64)

plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix4, annot=True, fmt='d', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted Problems')
plt.ylabel('Actual Problems')
plt.title('Problems Classification Model')
plt.show()
```



Conclusion:

the tuned logistic regression model #3 produces the highest scores, lets take a look in summary:

```
In [66]: scorestot = [('model1', acc1, recall1), ('model2', acc2, recall2), ('model3', acc3, recall3),
                    ('model3b', acc3b, recall3b), ('model3c', acc3c, recall3c), ('model4', acc4, recall4)]
for (name, x, y) in scorestot:
    print(name+' accuracy: '+str(x)+' ----- recall: '+str(y))
```

```
model1 accuracy: 1.0 ----- recall: 1.0
model2 accuracy: 0.722254720947663 ----- recall: 0.4767634552302272
model3 accuracy: 0.8801564813629965 ----- recall: 0.6980828574899769
model3a accuracy: 0.8478976340494937 ----- recall: 0.7358698608844331
model3b accuracy: 0.8801692777361387 ----- recall: 0.6982043494107641
model3c accuracy: 0.8801674496828328 ----- recall: 0.6981897703802697
model4 accuracy: 0.8791821289508802 ----- recall: 0.6788190985299477
```

```
In [67]: #using model 3b, attempt to increase recall
threshold = [0.45, 0.4, 0.35, 0.3, 0.25, 0.2]

for x in threshold:
    predictions = (prob3b[:, 1] > x).astype(int)
    # Calculate accuracy using the adjusted threshold
    accuracy_scores = accuracy_score(y_train3b, predictions)
    recall_scores = recall_score(y_train3b, predictions)
    precision_scores = precision_score(y_train3b, predictions)
    print("Accuracy with adjusted threshold of "+str(x)+" : {:.2f}".format(accuracy_scores))
    print("Recall with adjusted threshold of "+str(x)+" : {:.2f}".format(recall_scores))
    print("Precision with adjusted threshold of "+str(x)+" : {:.2f}".format(precision_scores))
    print('-----')
```

Accuracy with adjusted threshold of 0.45: 0.88

```
In [67]: #using model 3b, attempt to increase recall
threshold = [0.45, 0.4, 0.35, 0.3, 0.25, 0.2]

for x in threshold:
    predictions = (prob3b[:, 1] > x).astype(int)
    # Calculate accuracy using the adjusted threshold
    accuracy_scores = accuracy_score(y_train3b, predictions)
    recall_scores = recall_score(y_train3b, predictions)
    precision_scores = precision_score(y_train3b, predictions)
    print("Accuracy with adjusted threshold of "+str(x)+" : {:.2f}".format(accuracy_scores))
    print("Recall with adjusted threshold of "+str(x)+" : {:.2f}".format(recall_scores))
    print("Precision with adjusted threshold of "+str(x)+" : {:.2f}".format(precision_scores))
    print('-----')
```

```
Accuracy with adjusted threshold of 0.45: 0.88
Recall with adjusted threshold of 0.45: 0.71
Precision with adjusted threshold of 0.45: 0.96
-----
Accuracy with adjusted threshold of 0.4: 0.88
Recall with adjusted threshold of 0.4: 0.73
Precision with adjusted threshold of 0.4: 0.93
-----
Accuracy with adjusted threshold of 0.35: 0.87
Recall with adjusted threshold of 0.35: 0.75
Precision with adjusted threshold of 0.35: 0.90
-----
Accuracy with adjusted threshold of 0.3: 0.86
Recall with adjusted threshold of 0.3: 0.77
Precision with adjusted threshold of 0.3: 0.85
-----
Accuracy with adjusted threshold of 0.25: 0.84
Recall with adjusted threshold of 0.25: 0.81
Precision with adjusted threshold of 0.25: 0.78
-----
Accuracy with adjusted threshold of 0.2: 0.81
Recall with adjusted threshold of 0.2: 0.85
Precision with adjusted threshold of 0.2: 0.70
-----
```

Ok, lowering the threshold keeps accuracy relatively stable, and increases recall, but lowers precision. this is intuitive. So we are going to run the model one last time with a threshold of 0.35 to maintain an "A+" precision, while raising recall:

```
In [68]: #time to run through the test data!
# Predict the target class based on p > 0.5 criteria
y_pred3bFINAL = clf3b.predict(X_test_df3b)

# Predict the probability with the training data set
prob3bFINAL = clf3b.predict_proba(X_test_df3b)

# Calculate the model fit
acc3bFINAL = clf3b.score(X_test_df3b, y_test3b)

recall3bFINAL = recall_score(y_test3b, y_pred3bFINAL)
print('accuracy: '+str(acc3bFINAL)+' ----- recall: '+str(recall3bFINAL))
```

```
In [69]: #get final y_pred's on a 0.35 threshold:
accuracy: 0.87914418169321 ----- recall: 0.6960547426207643
y_pred3b_final = (prob3bFINAL[:, 1] > 0.35).astype(int)
conf_matrix3b_final = confusion_matrix(y_test3b, y_pred3b_final)

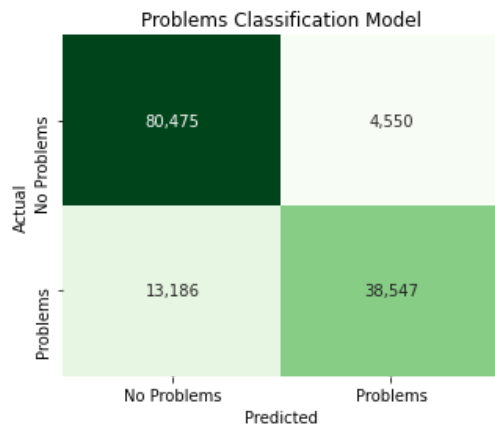
#creating the confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3b_final, annot=True, fmt=',', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Problems Classification Model')
plt.show()
```

```

In [69]: #get final y_pred's on a 0.35 threshold:
accuracy: 0.87914418169321 recall: 0.6960547426207643
y_pred3b_final = (prob3bFINAL[:, 1] > 0.35).astype(int)
conf_matrix3b_final = confusion_matrix(y_test3b, y_pred3b_final)

#creating the confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(conf_matrix3b_final, annot=True, fmt=',', cmap='Greens', cbar=False,
            xticklabels=['No Problems', 'Problems'],
            yticklabels=['No Problems', 'Problems'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Problems Classification Model')
plt.show()

```



```

In [70]: #find the column with the highest weight
X_train_df3b.columns[np.argmax(np.abs(clf3b.coef_))]

```

Out[70]: 'root_stone'

In []: