genetic_variant_classification

January 3, 2019

0.1 # Genetic Variant Classifications

0.2 Introduction

Accompanying slides:
https://docs.google.com/presentation/d/1UX6y4Z6ekvBifDRAp537xLp6KJlYovjIj7jO5LheU3o/edit?usp=s

The ClinVar dataset is a public resource containing annotations about human genetic variants. These variants are (usually manually) classified by clinical laboratories on a categorical spectrum ranging from benign, likely benign, uncertain significance, likely pathogenic, and pathogenic. Variants that have conflicting classifications (from laboratory to laboratory) can cause confusion when clinicians or researchers try to interpret whether the variant has an impact on the disease of a given patient.

Conflicting classifications are when two of any of the following three categories are present for one variant, two submissions of one category are not considered conflicting.

- Likely Benign or Benign
- VUS (uncertain significance)
- Likely Pathogenic or Pathogenic

0.3 Question

Given a set of variant features, we are going to try and identify whether that variant is likely to posses a conflicting classification or not using machine learning as opposed to manual classification.

Conflicting classification has been assigned to a CLASS column. It is a binary representation of whether or not a variant has conflicting classifications, where 0 represents consistent classifications and 1 represents conflicting classifications.

0.4 Models

The machine learning models I will be experimenting with to solve this problem are listed below:

- Logistic Regression
- Random Forest
- Gradient Boosting

```
In [207]: import pandas as pd
          import numpy as np
          import seaborn as sns
          import matplotlib.pyplot as plt
          from sklearn import ensemble
          from sklearn.model_selection import cross_val_score
          from sklearn.model_selection import train_test_split
          from sklearn.linear_model import LogisticRegression
          from imblearn.ensemble import EasyEnsemble
          from sklearn.metrics import roc_auc_score
          from sklearn.feature_selection import chi2
          from sklearn.preprocessing import Imputer
          from sklearn.feature_selection import chi2 as chi2_sk
          from sklearn.feature_selection import VarianceThreshold
          from sklearn.model_selection import GridSearchCV
          import scipy.stats as stats
          from sklearn.metrics import roc_curve
          from sklearn.metrics import confusion_matrix
          from sklearn.metrics import classification_report
          stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
          %matplotlib inline
```

0.5 Import the dataset

```
In [79]: df = pd.read_csv('./data/clinvar_conflicting.csv')
         df.shape
```

```
/Users/rook/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2698: Dtype
  interactivity=interactivity, compiler=compiler, result=result)
```

```
Out[79]: (65188, 46)
```

```
In [80]: df.head()
```

```
Out[80]:   CHROM     POS REF ALT  AF_ESP  AF_EXAC  AF_TGP  \
         0     1  955563   G   C  0.0000  0.00000  0.0000
         1     1  955597   G   T  0.0000  0.42418  0.2826
         2     1  955619   G   C  0.0000  0.03475  0.0088
         3     1  957640   C   T  0.0318  0.02016  0.0328
         4     1  976059   C   T  0.0000  0.00022  0.0010

                                        CLNDISDB CLNDISDBINCL  \
         0  MedGen:C3808739,OMIM:615120|MedGen:CN169374          NaN
         1                              MedGen:CN169374          NaN
         2  MedGen:C3808739,OMIM:615120|MedGen:CN169374          NaN
         3  MedGen:C3808739,OMIM:615120|MedGen:CN169374          NaN
         4                              MedGen:CN169374          NaN

                                                  CLNDN   ...    SIFT PolyPhen  \
         0  Myasthenic_syndrome,_congenital,_8|not_specified   ...     NaN      NaN
         1                                    not_specified   ...     NaN      NaN
```

```
2  Myasthenic_syndrome,_congenital,_8|not_specified   ...     NaN      NaN
3  Myasthenic_syndrome,_congenital,_8|not_specified   ...     NaN      NaN
4                                    not_specified   ...     NaN      NaN

  MOTIF_NAME MOTIF_POS HIGH_INF_POS MOTIF_SCORE_CHANGE  LoFtool  CADD_PHRED  \
0        NaN       NaN          NaN                NaN    0.421      11.390
1        NaN       NaN          NaN                NaN    0.421       8.150
2        NaN       NaN          NaN                NaN    0.421       3.288
3        NaN       NaN          NaN                NaN    0.421      12.560
4        NaN       NaN          NaN                NaN    0.421      17.740

   CADD_RAW BLOSUM62
0  1.133255     -2.0
1  0.599088      NaN
2  0.069819      1.0
3  1.356499      NaN
4  2.234711      NaN

[5 rows x 46 columns]
```

0.5.1 Binary class split

```
In [81]: df['CLASS'].value_counts()

Out[81]: 0    48754
         1    16434
         Name: CLASS, dtype: int64

In [82]: df['CLASS'].value_counts(normalize=True)

Out[82]: 0    0.747898
         1    0.252102
         Name: CLASS, dtype: float64
```

Slightly imbalanced. Accuracy will not be an effective metric for evaluating performance.

0.5.2 Data preprocessing

```
In [83]: X = df.loc[:, ~df.columns.isin(['CLASS'])]
         Y = df['CLASS']

In [84]: # Converting to categorical
         convert_cat = []
         to_drop = []
         unique = None

         categorical = X.select_dtypes(include=['object'])
         for i in categorical:
             column = categorical[i]
```

```
            print(i)
            unique = column.nunique()
            print(unique)

            cont = pd.crosstab(X[i], Y)
            chi2_res = scipy.stats.chi2_contingency(cont)

            # Keep all features with a significant P-value and drop the others
            if chi2_res[1] <= 0.05:
                convert_cat.append(i)
            else:
                to_drop.append(i)
```

```
CHROM
25
REF
866
ALT
458
CLNDISDB
9234
CLNDISDBINCL
48
CLNDN
9260
CLNDNINCL
54
CLNHGVS
65188
CLNSIGINCL
68
CLNVC
7
CLNVI
26289
MC
89
Allele
374
Consequence
48
IMPACT
4
SYMBOL
2328
Feature_type
2
Feature
```

```
2369
BIOTYPE
2
EXON
3264
INTRON
1929
cDNA_position
13970
CDS_position
13663
Protein_position
7339
Amino_acids
1262
Codons
2220
BAM_EDIT
2
SIFT
4
PolyPhen
4
MOTIF_NAME
2
HIGH_INF_POS
1
```

```
In [85]: print(f'convert_cat\n------\n{convert_cat} \n')
         print(f'to_drop\n------\n{to_drop}')
```

```
convert_cat
------
['CHROM', 'REF', 'ALT', 'CLNDISDB', 'CLNDN', 'CLNVC', 'MC', 'Allele', 'Consequence', 'IMPACT',

to_drop
------
['CLNDISDBINCL', 'CLNDNINCL', 'CLNHGVS', 'CLNSIGINCL', 'CLNVI', 'Feature_type', 'BIOTYPE', 'SI
```

0.5.3 Drop all features with more than 40% NaN's

```
In [86]: to_drop_nans = []

         for c in X.columns:
             if X[c].isnull().sum() / X.shape[0] > 0.40:
                 print(f'{c}: {X[c].isnull().sum() / X.shape[0]}')
```

```
                to_drop_nans.append(c)

                # Remove from the convert_cat array
                if c in convert_cat:
                    convert_cat.remove(c)

        print(f'\nNumber of features to drop: {len(to_drop_nans)}')
```

```
CLNDISDBINCL: 0.9988341412529913
CLNDNINCL: 0.9988341412529913
CLNSIGINCL: 0.9988341412529913
CLNVI: 0.5757041173222065
SSR: 0.9984046143461986
INTRON: 0.8649598085537216
DISTANCE: 0.998343253359514
BAM_EDIT: 0.509587654169479
SIFT: 0.6190096336749095
PolyPhen: 0.6196232435417561
MOTIF_NAME: 0.9999693195066577
MOTIF_POS: 0.9999693195066577
HIGH_INF_POS: 0.9999693195066577
MOTIF_SCORE_CHANGE: 0.9999693195066577
BLOSUM62: 0.6073970669448365

Number of features to drop: 15
```

```
In [87]: X = X.loc[:, ~X.columns.isin(to_drop_nans)]
```

There were a few features that were almost entirely made up of NaN values. Instead of imputing this data and essentially creating 60,000+ rows of made up figures, I decided to drop them all together.

```
In [88]: convert_cat
```

```
Out[88]: ['CHROM',
          'REF',
          'ALT',
          'CLNDISDB',
          'CLNDN',
          'CLNVC',
          'MC',
          'Allele',
          'Consequence',
          'IMPACT',
          'SYMBOL',
          'Feature',
          'EXON',
          'cDNA_position',
          'CDS_position',
          'Protein_position',
          'Amino_acids',
          'Codons']
```

Above are the remaining categorical features to be converted to binary form.

0.5.4 Convert features to categorical - get_dummies

```
In [89]: # Deleting duplicate rows
         X = X.loc[:,~X.columns.duplicated()]

         # Drop cols with too many unique values
         X = X.loc[:, ~X.columns.isin(to_drop)]

         # Get dummies - conver to categroical
         X = pd.get_dummies(data=X, columns=convert_cat)

         X = X.loc[:,~X.columns.duplicated()]
```

```
In [90]: X.shape
```

```
Out[90]: (65188, 66788)
```

```
In [91]: # Find all features created by get_dummies
         cat_feat = []

         for feature in convert_cat:
             for col in X.loc[:, X.columns.str.startswith(feature + "_")].columns:
                 cat_feat.append(col)
```

0.5.5 Run Chi2 after get_dummies

```
In [92]: gd_convert_cat = []
         gd_to_dropt = []

         for feature in cat_feat:
             cont = pd.crosstab(X[feature], Y)
             chi2_res = scipy.stats.chi2_contingency(cont)

             # Keep all features with a significant P-value and drop the others
             if chi2_res[1] <= 0.05:
                 gd_convert_cat.append(feature)
             else:
                 gd_to_dropt.append(feature)
```

```
In [93]: len(gd_to_dropt)
```

```
Out[93]: 64912
```

```
In [112]: # Get all categoircal features again
          # Find all features created by get_dummies
          cat_feat = []

          for feature in convert_cat:
              for col in X.loc[:, X.columns.str.startswith(feature + "_")].columns:
                  cat_feat.append(col)
```

We can see that the results of the Chi2 test are show that we should drop a substantial number of dummy features.

```
In [94]: X = X.loc[:, ~X.columns.isin(gd_to_dropt)]

In [95]: X.shape

Out[95]: (65188, 1876)
```

Above we can see the number of remaining features by examining the shape of X.

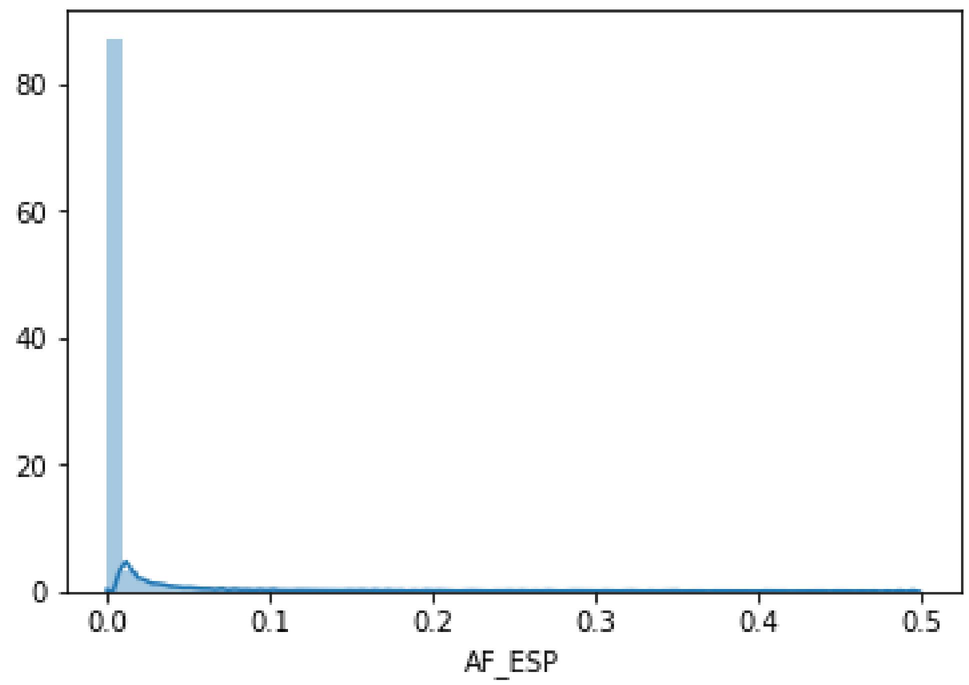
0.5.6 Finding all NaN rows

```
In [115]: nans = lambda X: X[X.isnull().any(axis=1)]
          len(nans(X))

Out[115]: 10940
```

There are still quite a few rows with NaN values. We will have to impute the numerical features.

0.5.7 Numerical Imputer

```
In [114]: numerical_data = X.loc[:, ~X.columns.isin(X[cat_feat])]
          print(numerical_data.columns)

Index(['POS', 'AF_ESP', 'AF_EXAC', 'AF_TGP', 'ORIGIN', 'STRAND', 'LoFtool',
       'CADD_PHRED', 'CADD_RAW'],
      dtype='object')


In [116]: sns.distplot(X['AF_ESP'])

Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x1a45f25ba8>
```



We can see that these numerical features are not normally distributed. So we will use the median to impute the data.

```
In [117]: # For numerical data, impute using mean OR median values
          imp = Imputer(missing_values='NaN', strategy='median', axis=0)
          imp = imp.fit(numerical_data)

          # Impute our data
          X[numerical_data.columns] = imp.transform(numerical_data)

In [118]: nans = lambda X: X[X.isnull().any(axis=1)]
          len(nans(X))

Out[118]: 0
```

We can see that we no longer have any NaN values in our data.
We do not need to impute the categorical features because get_dummies does this for us.

0.5.8 VarianceThreshold

```
In [119]: # Removes all low-variance features
          def variance_threshold_selector(data, threshold=0.05):
              selector = VarianceThreshold(threshold)
              selector.fit(data)
              return data[data.columns[selector.get_support(indices=True)]]
```

```
         vt_to_keep = variance_threshold_selector(X)

         vt_to_keep.head()

Out[119]:          POS  ORIGIN  STRAND  LoFtool  CADD_PHRED  CADD_RAW  CHROM_1  CHROM_2  \
         0  955563.0     1.0     1.0    0.421      11.390  1.133255        1        0
         1  955597.0     1.0     1.0    0.421       8.150  0.599088        1        0
         2  955619.0     1.0     1.0    0.421       3.288  0.069819        1        0
         3  957640.0     1.0     1.0    0.421      12.560  1.356499        1        0
         4  976059.0     1.0     1.0    0.421      17.740  2.234711        1        0

            CHROM_5  CHROM_11        ...         MC_SO:0001627|intron_variant  Allele_A  \
         0        0         0        ...                                    0         0
         1        0         0        ...                                    0         0
         2        0         0        ...                                    0         0
         3        0         0        ...                                    0         0
         4        0         0        ...                                    0         0

            Allele_C  Allele_G  Allele_T  Consequence_missense_variant  IMPACT_HIGH  \
         0         1         0         0                             1            0
         1         0         0         1                             0            0
         2         1         0         0                             1            0
         3         0         0         1                             0            0
         4         0         0         1                             0            0

            IMPACT_LOW  IMPACT_MODERATE  IMPACT_MODIFIER
         0           0                1                0
         1           1                0                0
         2           0                1                0
         3           1                0                0
         4           1                0                0

         [5 rows x 33 columns]
```

```
In [122]: X = X[vt_to_keep.columns]
```

Variables with low variance really aren't beneficial to our model.

0.5.9 Display correlation Matrix to identify features that need to be dropped

```
In [124]: correlation_matrix = X.corr()
          display(correlation_matrix)
```

```
                                               POS    ORIGIN    STRAND   LoFtool  \
POS                                       1.000000  0.010341 -0.122584  0.258381
ORIGIN                                    0.010341  1.000000 -0.009006 -0.021575
STRAND                                   -0.122584 -0.009006  1.000000 -0.148790
LoFtool                                   0.258381 -0.021575 -0.148790  1.000000
CADD_PHRED                               -0.004949  0.046219  0.014293 -0.032411
```

```
CADD_RAW                         -0.008799  0.052163  0.009340 -0.033348
CHROM_1                           0.199746  0.001611 -0.037417 -0.042282
CHROM_2                           0.403622 -0.000724 -0.122106  0.186457
CHROM_5                           0.109581 -0.000823  0.117946  0.015560
CHROM_11                         -0.006367 -0.006612  0.083352  0.121785
CHROM_17                         -0.189223 -0.007932 -0.083800 -0.091613
REF_A                             0.008344 -0.002620  0.084427  0.022992
REF_C                            -0.005175 -0.003326  0.006302 -0.022567
REF_G                            -0.008914  0.004618 -0.030990 -0.032604
REF_T                             0.023565 -0.007661 -0.051144  0.053027
ALT_A                            -0.001230  0.006929 -0.037090 -0.031078
ALT_C                             0.004383 -0.005351 -0.041008  0.037114
ALT_G                            -0.001900 -0.003378  0.074429  0.017705
ALT_T                             0.002194 -0.001144  0.007518 -0.013454
CLNDISDB_MedGen:CN169374          0.054812 -0.013733 -0.031498  0.033038
CLNDN_not_specified               0.054812 -0.013733 -0.031498  0.033038
CLNVC_single_nucleotide_variant   0.022069 -0.016250  0.001431  0.004151
MC_SO:0001583|missense_variant    0.008146  0.020126  0.034285  0.046276
MC_SO:0001627|intron_variant     -0.004403 -0.012429 -0.002026 -0.018733
Allele_A                          0.000109  0.005893 -0.038894 -0.029711
Allele_C                          0.009060 -0.008518 -0.041034  0.039009
Allele_G                         -0.001598 -0.004381  0.076873  0.017905
Allele_T                          0.003403 -0.002587  0.009328 -0.013260
Consequence_missense_variant      0.005505  0.008358 -0.000550  0.029406
IMPACT_HIGH                      -0.031125  0.044327 -0.000787 -0.004316
IMPACT_LOW                        0.021021 -0.028777  0.002695 -0.009589
IMPACT_MODERATE                   0.000822  0.010887  0.000314  0.026423
IMPACT_MODIFIER                  -0.007929 -0.012201 -0.004366 -0.027059

                                  CADD_PHRED  CADD_RAW   CHROM_1   CHROM_2  \
POS                                -0.004949 -0.008799  0.199746  0.403622
ORIGIN                              0.046219  0.052163  0.001611 -0.000724
STRAND                              0.014293  0.009340 -0.037417 -0.122106
LoFtool                            -0.032411 -0.033348 -0.042282  0.186457
CADD_PHRED                          1.000000  0.954832  0.013997  0.003979
CADD_RAW                            0.954832  1.000000  0.013964 -0.004036
CHROM_1                             0.013997  0.013964  1.000000 -0.105889
CHROM_2                             0.003979 -0.004036 -0.105889  1.000000
CHROM_5                            -0.001933  0.000534 -0.065886 -0.095132
CHROM_11                            0.043881  0.047204 -0.076743 -0.110809
CHROM_17                            0.013041  0.012929 -0.081337 -0.117441
REF_A                              -0.139935 -0.125550 -0.018791  0.009397
REF_C                               0.166312  0.130640  0.012841 -0.003046
REF_G                               0.023686  0.038259  0.006412 -0.013088
REF_T                              -0.147904 -0.129229 -0.003924  0.022848
ALT_A                               0.067779  0.074191  0.007426 -0.013449
ALT_C                              -0.150129 -0.131645 -0.001742  0.009793
ALT_G                              -0.111181 -0.107058 -0.022122  0.012520
```

```
ALT_T                              0.143910  0.119086  0.013182 -0.005322
CLNDISDB_MedGen:CN169374          -0.090316 -0.086369  0.020139  0.014228
CLNDN_not_specified               -0.090316 -0.086369  0.020139  0.014228
CLNVC_single_nucleotide_variant   -0.060210 -0.054330  0.004341  0.014317
MC_SO:0001583|missense_variant     0.281877  0.208015 -0.017147  0.031349
MC_SO:0001627|intron_variant      -0.289163 -0.243923  0.015639 -0.028737
Allele_A                           0.063055  0.070973  0.007160 -0.011292
Allele_C                          -0.166418 -0.146230 -0.001187  0.014122
Allele_G                          -0.113621 -0.109007 -0.021083  0.012929
Allele_T                           0.139637  0.115297  0.012936 -0.004231
Consequence_missense_variant       0.351695  0.262472 -0.005649  0.038108
IMPACT_HIGH                        0.376313  0.470595 -0.007176 -0.020912
IMPACT_LOW                        -0.447707 -0.422492  0.007269 -0.015183
IMPACT_MODERATE                    0.366433  0.269884 -0.007203  0.036511
IMPACT_MODIFIER                   -0.250787 -0.208329  0.007302 -0.020248

                                  CHROM_5  CHROM_11       ...          \
POS                              0.109581 -0.006367       ...
ORIGIN                          -0.000823 -0.006612       ...
STRAND                           0.117946  0.083352       ...
LoFtool                          0.015560  0.121785       ...
CADD_PHRED                      -0.001933  0.043881       ...
CADD_RAW                         0.000534  0.047204       ...
CHROM_1                         -0.065886 -0.076743       ...
CHROM_2                         -0.095132 -0.110809       ...
CHROM_5                          1.000000 -0.068947       ...
CHROM_11                        -0.068947  1.000000       ...
CHROM_17                        -0.073073 -0.085115       ...
REF_A                            0.047899  0.019456       ...
REF_C                           -0.018144 -0.011086       ...
REF_G                           -0.025718 -0.011207       ...
REF_T                            0.008074 -0.002053       ...
ALT_A                           -0.018794 -0.012647       ...
ALT_C                           -0.001353  0.005288       ...
ALT_G                            0.043140  0.017194       ...
ALT_T                           -0.013616 -0.009985       ...
CLNDISDB_MedGen:CN169374        -0.008678 -0.028621       ...
CLNDN_not_specified             -0.008678 -0.028621       ...
CLNVC_single_nucleotide_variant  0.000377 -0.026252       ...
MC_SO:0001583|missense_variant   0.033218  0.011094       ...
MC_SO:0001627|intron_variant    -0.020896 -0.005134       ...
Allele_A                        -0.019389 -0.014104       ...
Allele_C                        -0.000967  0.000164       ...
Allele_G                         0.042566  0.017053       ...
Allele_T                        -0.015235 -0.009525       ...
Consequence_missense_variant     0.011463 -0.010358       ...
IMPACT_HIGH                      0.004223  0.059553       ...
IMPACT_LOW                      -0.000490 -0.020470       ...
```

```
IMPACT_MODERATE                  0.011217 -0.008884       ...
IMPACT_MODIFIER                 -0.023139 -0.005007       ...

                                 MC_SO:0001627|intron_variant  Allele_A  \
POS                                                 -0.004403  0.000109
ORIGIN                                              -0.012429  0.005893
STRAND                                              -0.002026 -0.038894
LoFtool                                             -0.018733 -0.029711
CADD_PHRED                                          -0.289163  0.063055
CADD_RAW                                            -0.243923  0.070973
CHROM_1                                              0.015639  0.007160
CHROM_2                                             -0.028737 -0.011292
CHROM_5                                             -0.020896 -0.019389
CHROM_11                                            -0.005134 -0.014104
CHROM_17                                             0.005155 -0.009609
REF_A                                                0.017176 -0.278573
REF_C                                               -0.036336 -0.295403
REF_G                                               -0.034630  0.682969
REF_T                                                0.035914 -0.152598
ALT_A                                               -0.026992  0.968190
ALT_C                                                0.025504 -0.304537
ALT_G                                                0.021112 -0.310225
ALT_T                                               -0.031116 -0.445898
CLNDISDB_MedGen:CN169374                             0.111829 -0.000629
CLNDN_not_specified                                  0.111829 -0.000629
CLNVC_single_nucleotide_variant                     -0.099950  0.132200
MC_SO:0001583|missense_variant                      -0.267217 -0.013278
MC_SO:0001627|intron_variant                         1.000000 -0.029147
Allele_A                                            -0.029147  1.000000
Allele_C                                             0.020332 -0.293685
Allele_G                                             0.012103 -0.304020
Allele_T                                            -0.035320 -0.439263
Consequence_missense_variant                        -0.348666 -0.008659
IMPACT_HIGH                                         -0.093798 -0.024169
IMPACT_LOW                                           0.085153  0.050329
IMPACT_MODERATE                                     -0.361405 -0.025003
IMPACT_MODIFIER                                      0.589508 -0.017579

                                 Allele_C  Allele_G  Allele_T  \
POS                              0.009060 -0.001598  0.003403
ORIGIN                          -0.008518 -0.004381 -0.002587
STRAND                          -0.041034  0.076873  0.009328
LoFtool                          0.039009  0.017905 -0.013260
CADD_PHRED                      -0.166418 -0.113621  0.139637
CADD_RAW                        -0.146230 -0.109007  0.115297
CHROM_1                         -0.001187 -0.021083  0.012936
CHROM_2                          0.014122  0.012929 -0.004231
CHROM_5                         -0.000967  0.042566 -0.015235
```

```
CHROM_11                        0.000164  0.017053 -0.009525
CHROM_17                        0.027534 -0.000740 -0.016322
REF_A                          -0.008791  0.587959 -0.164275
REF_C                          -0.315161 -0.074736  0.677756
REF_G                          -0.073262 -0.321352 -0.291857
REF_T                           0.582153 -0.014944 -0.273352
ALT_A                          -0.298006 -0.308493 -0.445726
ALT_C                           0.953931 -0.212236 -0.306649
ALT_G                          -0.208850  0.969924 -0.312375
ALT_T                          -0.300188 -0.310752  0.970628
CLNDISDB_MedGen:CN169374        0.002535 -0.006228  0.010208
CLNDN_not_specified             0.002535 -0.006228  0.010208
CLNVC_single_nucleotide_variant  0.093995  0.098529  0.138074
MC_SO:0001583|missense_variant   0.030828  0.063529  0.008090
MC_SO:0001627|intron_variant     0.020332  0.012103 -0.035320
Allele_A                       -0.293685 -0.304020 -0.439263
Allele_C                        1.000000 -0.204673 -0.295721
Allele_G                       -0.204673  1.000000 -0.306128
Allele_T                       -0.295721 -0.306128  1.000000
Consequence_missense_variant    0.051132  0.066565  0.008400
IMPACT_HIGH                    -0.070006 -0.076095 -0.027678
IMPACT_LOW                     -0.017249 -0.021332  0.037167
IMPACT_MODERATE                 0.042277  0.056105 -0.007122
IMPACT_MODIFIER                 0.018539  0.006363 -0.024113

                                Consequence_missense_variant  IMPACT_HIGH  \
POS                                                 0.005505    -0.031125
ORIGIN                                              0.008358     0.044327
STRAND                                             -0.000550    -0.000787
LoFtool                                             0.029406    -0.004316
CADD_PHRED                                          0.351695     0.376313
CADD_RAW                                            0.262472     0.470595
CHROM_1                                            -0.005649    -0.007176
CHROM_2                                             0.038108    -0.020912
CHROM_5                                             0.011463     0.004223
CHROM_11                                           -0.010358     0.059553
CHROM_17                                            0.013835     0.022232
REF_A                                               0.046655    -0.059272
REF_C                                               0.023462    -0.038900
REF_G                                               0.006824    -0.035849
REF_T                                               0.017436    -0.045607
ALT_A                                              -0.021578     0.000526
ALT_C                                               0.023908    -0.016626
ALT_G                                               0.050888    -0.053652
ALT_T                                              -0.004995    -0.010021
CLNDISDB_MedGen:CN169374                           -0.048313    -0.076266
CLNDN_not_specified                                -0.048313    -0.076266
CLNVC_single_nucleotide_variant                     0.227704    -0.425647
```

```
MC_SO:0001583|missense_variant                             0.713036    -0.197650
MC_SO:0001627|intron_variant                              -0.348666    -0.093798
Allele_A                                                  -0.008659    -0.024169
Allele_C                                                   0.051132    -0.070006
Allele_G                                                   0.066565    -0.076095
Allele_T                                                   0.008400    -0.027678
Consequence_missense_variant                               1.000000    -0.270683
IMPACT_HIGH                                               -0.270683     1.000000
IMPACT_LOW                                                -0.680527    -0.197681
IMPACT_MODERATE                                            0.947186    -0.285776
IMPACT_MODIFIER                                           -0.295407    -0.085810

                                 IMPACT_LOW  IMPACT_MODERATE  IMPACT_MODIFIER
POS                                0.021021         0.000822        -0.007929
ORIGIN                            -0.028777         0.010887        -0.012201
STRAND                             0.002695         0.000314        -0.004366
LoFtool                           -0.009589         0.026423        -0.027059
CADD_PHRED                        -0.447707         0.366433        -0.250787
CADD_RAW                          -0.422492         0.269884        -0.208329
CHROM_1                            0.007269        -0.007203         0.007302
CHROM_2                           -0.015183         0.036511        -0.020248
CHROM_5                           -0.000490         0.011217        -0.023139
CHROM_11                          -0.020470        -0.008884        -0.005007
CHROM_17                          -0.029651         0.016024         0.000620
REF_A                             -0.014781         0.040378         0.007805
REF_C                              0.025768         0.009259        -0.023769
REF_G                              0.034373        -0.004315        -0.016834
REF_T                             -0.002753         0.011711         0.026083
ALT_A                              0.040083        -0.028601        -0.016850
ALT_C                             -0.032670         0.026643         0.022830
ALT_G                             -0.028659         0.049386         0.009847
ALT_T                              0.031563        -0.010454        -0.025134
CLNDISDB_MedGen:CN169374           0.011022        -0.048396         0.138769
CLNDN_not_specified                0.011022        -0.048396         0.138769
CLNVC_single_nucleotide_variant    0.126268         0.136975        -0.061770
MC_SO:0001583|missense_variant    -0.519487         0.717232        -0.223459
MC_SO:0001627|intron_variant       0.085153        -0.361405         0.589508
Allele_A                           0.050329        -0.025003        -0.017579
Allele_C                          -0.017249         0.042277         0.018539
Allele_G                          -0.021332         0.056105         0.006363
Allele_T                           0.037167        -0.007122        -0.024113
Consequence_missense_variant      -0.680527         0.947186        -0.295407
IMPACT_HIGH                       -0.197681        -0.285776        -0.085810
IMPACT_LOW                         1.000000        -0.718472        -0.215737
IMPACT_MODERATE                   -0.718472         1.000000        -0.311878
IMPACT_MODIFIER                   -0.215737        -0.311878         1.000000

[33 rows x 33 columns]
```

```
In [125]: # Correlated features to be dropped
          upper = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape), k=1).asty
          to_drop = [column for column in upper.columns if any(upper[column] > 0.90)]
          print(f'Number of correlated features to drop: {len(to_drop)}')
```

```
Number of correlated features to drop: 7
```

```
In [126]: cols = list(X.columns)

          for col in to_drop:
              cols.remove(col)

          len(cols)

          X = X[cols]
```

Here we are dropping highly correlated features.

```
In [127]: # Save the DF for future reference
          X.to_csv('./data/X_df_v2.csv')
```

0.6 Train / Test splits

```
In [128]: # Train splits
          X_train, X_temp, y_train, y_temp = train_test_split(X, Y, test_size=0.40, random_sta

          # Dev and Test splits
          X_dev, X_test, y_dev, y_test = train_test_split(X_temp, y_temp, test_size=0.50, rando
```

We are splitting the data into a train, test and dev set.

```
In [223]: # Function for plotting the ROC curve
          def plot_roc_curve(labels, probs):
              fpr, tpr, thresholds = roc_curve(labels, probs)

              plt.plot(fpr, tpr, label="ROC Curve")
              plt.xlabel("FPR")
              plt.ylabel("TPR (recall)")
              # find threshold closest to zero
              close_zero = np.argmin(np.abs(thresholds))
              plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
                          label="threshold zero", fillstyle="none", c='k', mew=2)
              plt.legend(loc=4)
```

0.7 Logistic Regression

```
In [239]: # Define the LR model
          lr = LogisticRegression()
          lr.fit(X_train, y_train)

          # Train
          lr_preds_train = lr.predict_proba(X_train)
          lr_probs_train = lr_preds_train[:, 1]
          lr_roc_train = roc_auc_score(y_train, lr_probs_train)

          # Dev
          lr_preds_dev = lr.predict_proba(X_dev)
          lr_probs_dev = lr_preds_dev[:, 1]
          lr_roc_dev = roc_auc_score(y_dev, lr_probs_dev)

          print(f'roc_auc_score Train: {lr_roc_train}')
          print(f'roc_auc_score Dev: {lr_roc_dev}\n')

          probs_lr = np.argmax(lr_preds_dev, axis=1)
          conf_matrix_lr_dev = confusion_matrix(y_dev, probs_lr)
          print(f'Confusion Matrix:\n{conf_matrix_lr_dev}\n')

          print(classification_report(y_dev, np.argmax(lr_preds_dev, axis=1), target_names=['N
```

```
roc_auc_score Train: 0.4988218485572249
roc_auc_score Dev: 0.49571504168427244

Confusion Matrix:
[[9751    0]
 [3287    0]]

                 precision    recall  f1-score   support

Non-Conflicting       0.75      1.00      0.86      9751
    Conflicting       0.00      0.00      0.00      3287

    avg / total       0.56      0.75      0.64     13038
```

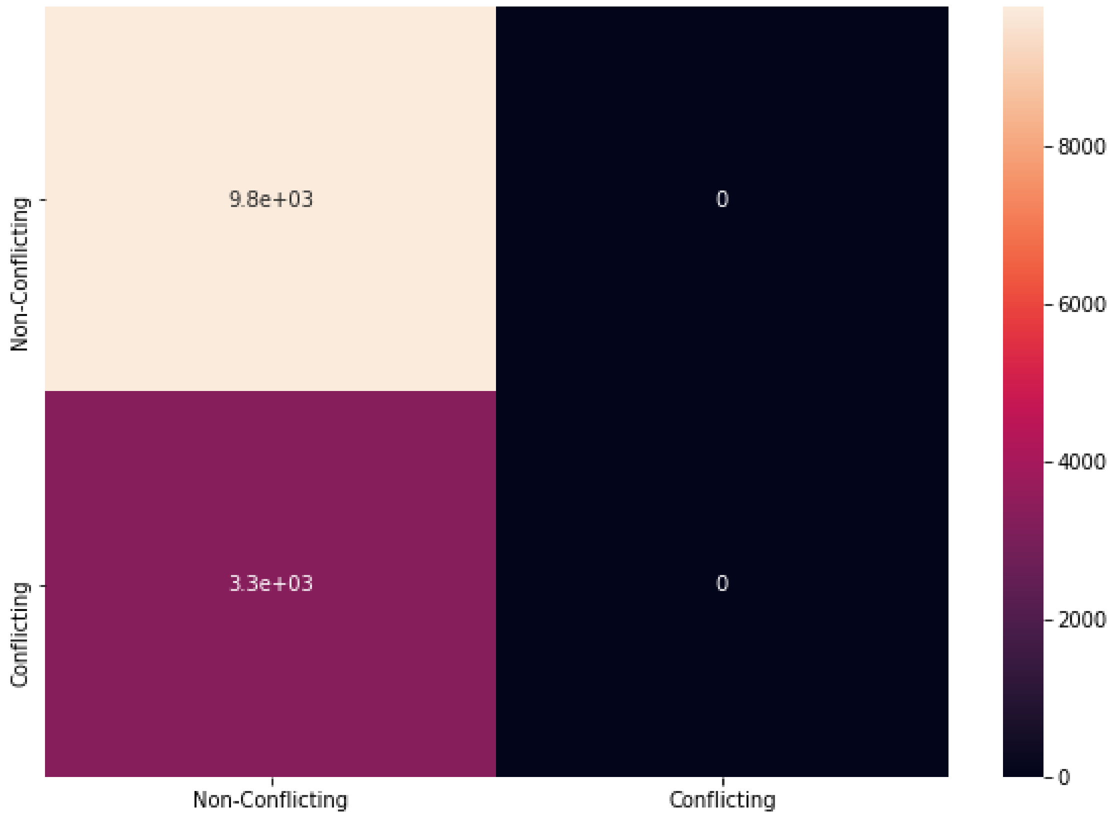
```
/Users/rook/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Unde
  'precision', 'predicted', average, warn_for)
```

From these results, it would appear that this is a pretty useless model.

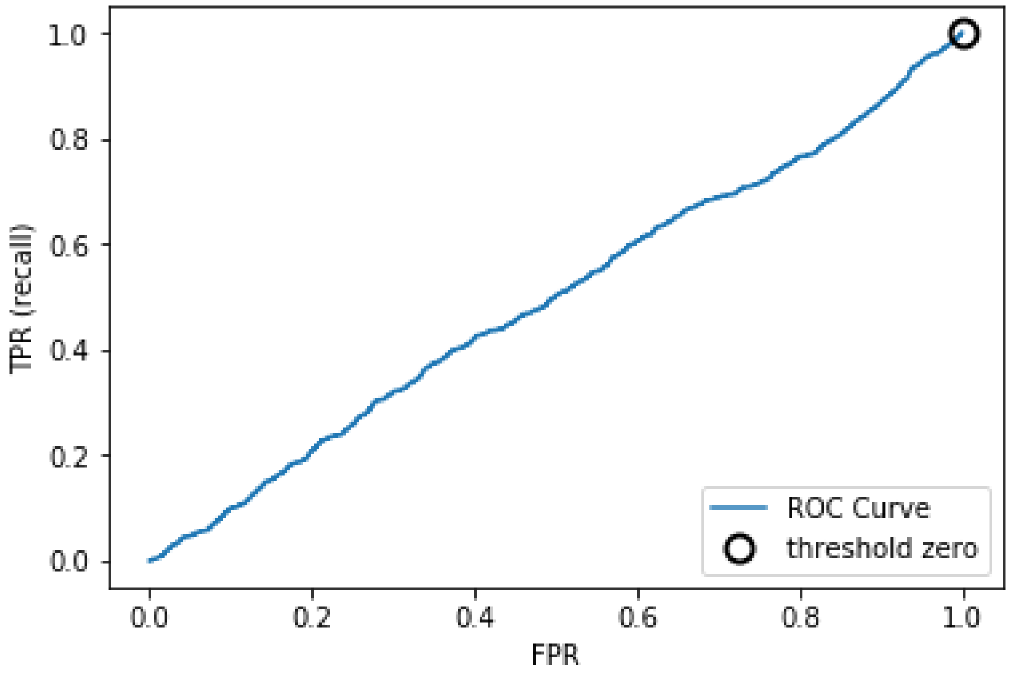
0.8 Logistic Regression Grid Search

```
In [260]: lr_gs = LogisticRegression()

          # Create regularization penalty space
          penalty = ['l1', 'l2']

          # Create regularization hyperparameter space
          C = np.logspace(0, 4, 10)

          # Create hyperparameter options
          hyperparameters = dict(C=C, penalty=penalty)

          grid_log_r = GridSearchCV(lr_gs, param_grid=hyperparameters, refit='recall_score', c
          grid_log_r.fit(X_train, y_train)

          # Train
          lr_preds_train = grid_log_r.predict_proba(X_train)
          lr_probs_train = lr_preds_train[:, 1]
          lr_roc_train = roc_auc_score(y_train, lr_probs_train)

          # Dev
          lr_preds_dev = grid_log_r.predict_proba(X_dev)
          lr_probs_dev = lr_preds_dev[:, 1]
          lr_roc_dev = roc_auc_score(y_dev, lr_probs_dev)

          # Confusion matrix
          conf_matrix_lr_dev = confusion_matrix(y_dev, np.argmax(lr_preds_dev, axis=1))

          # Best parameters from the grid search
          best_params = grid_log_r.best_params_
          print(f'Best Parameters: {best_params}\n')

          print(f'roc_auc_score Train: {lr_roc_train}')
          print(f'roc_auc_score Dev: {lr_roc_dev}\n')
          print(f'Confusion Matrix:\n{conf_matrix_lr_dev} \n')

          print(classification_report(y_dev, np.argmax(lr_preds_dev, axis=1), target_names=['N

          # Classification report and heatmap
          df_cm = pd.DataFrame(conf_matrix_lr_dev, index = [i for i in ['Non-Conflicting', 'Co
                            columns = ['Non-Conflicting', 'Conflicting'])

          plt.figure(figsize = (10,7))
          sns.heatmap(df_cm, annot=True)
```

```
Best Parameters: {'C': 1.0, 'penalty': 'l2'}

roc_auc_score Train: 0.4988218485572249
roc_auc_score Dev: 0.49571504168427244

Confusion Matrix:
[[9751    0]
 [3287    0]]

                 precision    recall  f1-score   support

Non-Conflicting       0.75      1.00      0.86      9751
    Conflicting       0.00      0.00      0.00      3287

    avg / total       0.56      0.75      0.64     13038



/Users/rook/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: Unde
  'precision', 'predicted', average, warn_for)


Out[260]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2018bdd8>
```



```
In [241]: # Plot ROC curve
          plot_roc_curve(y_dev, lr_probs_dev)
```



With this model, the AUC is roughly at chance level, meaning that the output is as good as random.

0.9 Random Forest

```
In [253]: rfc = ensemble.RandomForestClassifier(n_estimators=100, n_jobs=-1)
          rfc.fit(X_train, y_train)

          # Train
          rfc_preds_train = rfc.predict_proba(X_train)
          rfc_probs_train = rfc_preds_train[:, 1]
          rfc_roc_train = roc_auc_score(y_train, rfc_probs_train)

          # Dev
          rfc_preds_dev = rfc.predict_proba(X_dev)
          rfc_probs_dev = rfc_preds_dev[:, 1]
          rfc_roc_dev = roc_auc_score(y_dev, rfc_probs_dev)

          # AUC Scores
          print(f'roc_auc_score Train: {rfc_roc_train}')
          print(f'roc_auc_score Dev: {rfc_roc_dev}\n')
```

```
    # Confusion Matrix
    conf_matrix_rf_dev = confusion_matrix(y_dev, np.argmax(rfc_preds_dev, axis=1))
    print(f'Confusion Matrix:\n{conf_matrix_rf_dev}\n')

    # Classification Report
    print(classification_report(y_dev, np.argmax(rfc_preds_dev, axis=1), target_names=['
```

```
roc_auc_score Train: 1.0
roc_auc_score Dev: 0.6322950596721774

Confusion Matrix:
[[8854  897]
 [2700  587]]

                 precision    recall  f1-score   support

Non-Conflicting       0.77      0.91      0.83      9751
    Conflicting       0.40      0.18      0.25      3287

    avg / total       0.67      0.72      0.68     13038
```

This is a big improvement compared to the previous Logistic Regression model. TP's have increased from 0 to 587.

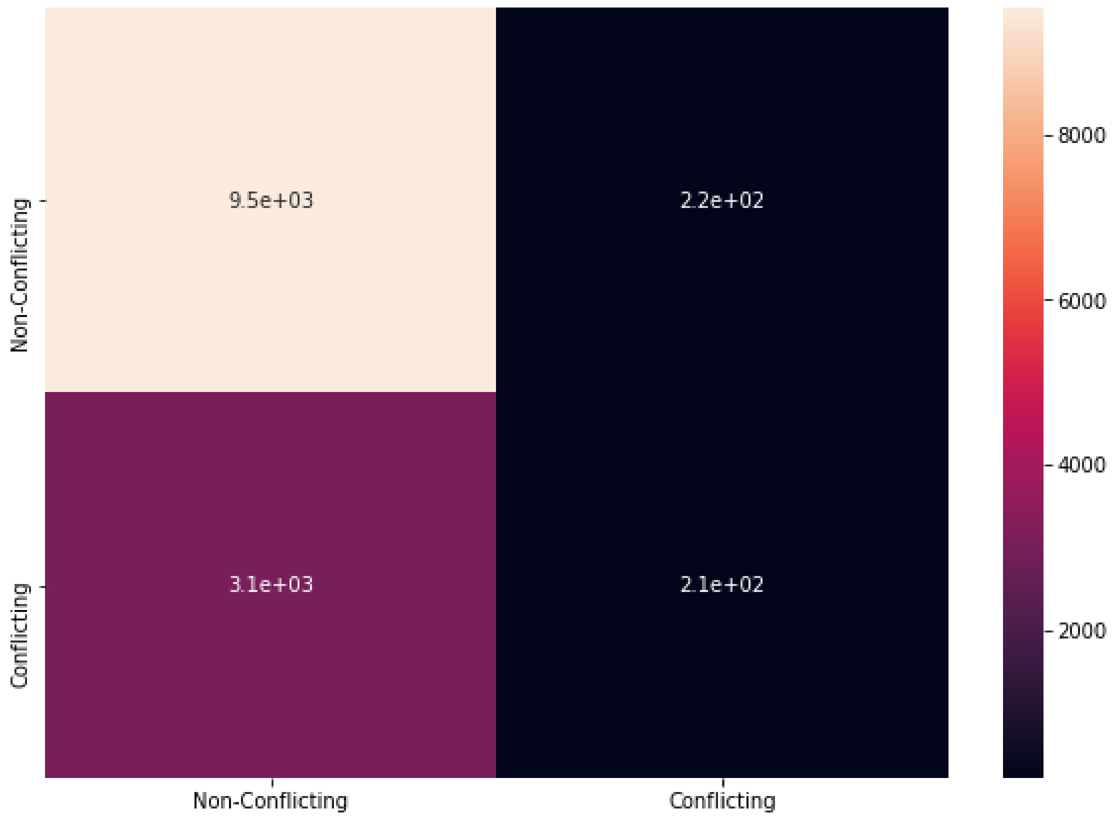
0.10 Random Forest Grid Search

```
In [262]: rfc = ensemble.RandomForestClassifier()

          parameters = {'n_estimators':[100,300,500],
                        'max_features':['sqrt', 'log2'],
                        'min_samples_split':[2,8,20]
                       }

          # Instantiating and fitting Grid Search, then printing best score and best parameter
          grid_rfc = GridSearchCV(rfc, param_grid=parameters, refit='recall_score', cv=5, n_jo
          grid_rfc.fit(X_train, y_train)

          # Train
          rfc_preds_train = grid_rfc.predict_proba(X_train)
          rfc_probs_train = rfc_preds_train[:, 1]
          rfc_roc_train = roc_auc_score(y_train, rfc_probs_train)

          # Dev
          rfc_preds_dev = grid_rfc.predict_proba(X_dev)
          rfc_probs_dev = rfc_preds_dev[:, 1]
          rfc_roc_dev = roc_auc_score(y_dev, rfc_probs_dev)
```

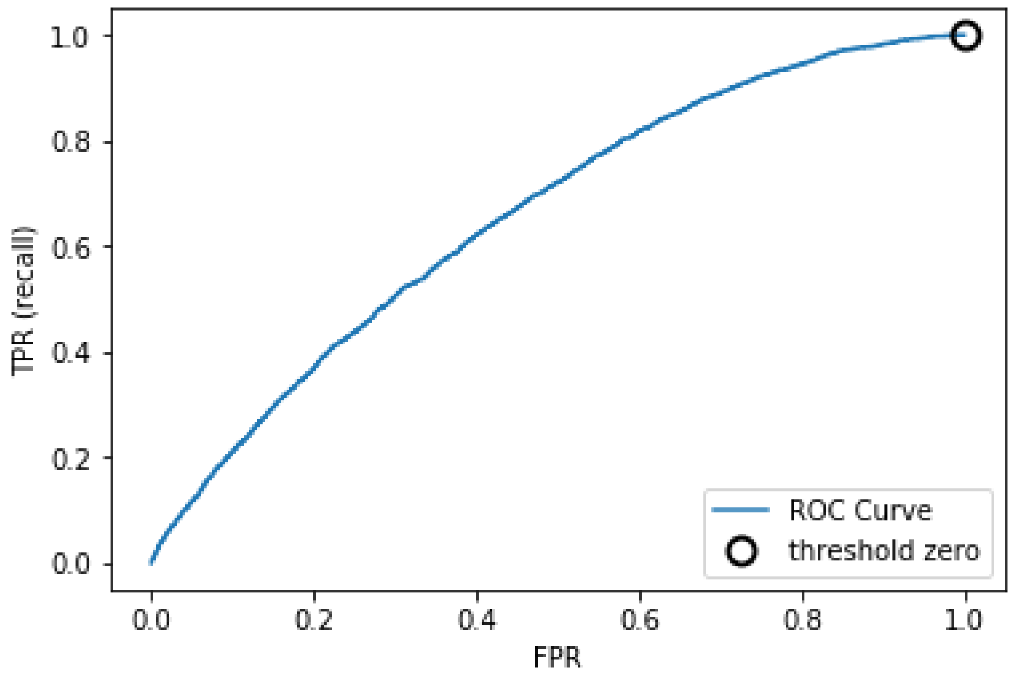
```
# Confusion matrix
conf_matrix_rf_dev = confusion_matrix(y_dev, np.argmax(rfc_preds_dev, axis=1))

# Best parameters from the grid search
best_params = grid_rfc.best_params_
print(f'Best Parameters: {best_params}\n')

print(f'roc_auc_score Train: {rfc_roc_train}')
print(f'roc_auc_score Dev: {rfc_roc_dev}\n')
print(f'Confusion Matrix: \n {conf_matrix_rf_dev} \n')
print(classification_report(y_dev, np.argmax(rfc_preds_dev, axis=1), target_names=['I

# Classification report and heatmap
df_cm = pd.DataFrame(conf_matrix_rf_dev, index = [i for i in ['Non-Conflicting', 'Co
                  columns = ['Non-Conflicting', 'Conflicting'])

plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True)
```

```
Best Parameters: {'max_features': 'sqrt', 'min_samples_split': 20, 'n_estimators': 300}

roc_auc_score Train: 0.9447069030698894
roc_auc_score Dev: 0.6588164555103864

Confusion Matrix:
 [[9530  221]
 [3078  209]]

                 precision    recall  f1-score   support

Non-Conflicting       0.76      0.98      0.85      9751
    Conflicting       0.49      0.06      0.11      3287

    avg / total       0.69      0.75      0.67     13038
```

```
Out[262]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3e5ff400>
```



```
In [263]: # Plot ROC curve
          plot_roc_curve(y_dev, rfc_probs_dev)
```



For some reason, the model worsened after performing grid search. I was especially surprised to see that recall worsened considering I was optimizing for recall with, refit='recall_score'.

0.11 Gradient Boosting

```
In [250]: # We'll make 500 iterations, use 2-deep trees, and set our loss function.
          params = {'n_estimators': 700,
                    'max_depth': 3,
                    'loss': 'deviance'}

          # Initialize and fit the model.
          gb_clf = ensemble.GradientBoostingClassifier(**params)
          gb_clf.fit(X_train, y_train)

          # Train
          gb_preds_train = gb_clf.predict_proba(X_train)
          gb_probs_train = gb_preds_train[:, 1]
          gb_roc_train = roc_auc_score(y_train, gb_probs_train)

          # Dev
          gb_preds_dev = gb_clf.predict_proba(X_dev)
          gb_probs_dev = gb_preds_dev[:, 1]
          gb_roc_dev = roc_auc_score(y_dev, gb_probs_dev)

          # AUC Scores
```

```
        print(f'roc_auc_score Train: {gb_roc_train}')
        print(f'roc_auc_score Dev: {gb_roc_dev}\n')

        # Confusion Matrix
        probs_gb = np.argmax(gb_preds_dev, axis=1)
        conf_matrix_gb_dev = confusion_matrix(y_dev, probs_gb)
        print(f'Confusion Matrix: \n {conf_matrix_gb_dev}\n')

        # Classification Report
        print(classification_report(y_dev, probs_gb, target_names=['Non-Conflicting', 'Confli
```

```
roc_auc_score Train: 0.7514867050924068
roc_auc_score Dev: 0.6690795826733675

Confusion Matrix:
 [[9610  141]
 [3139  148]]

                 precision    recall  f1-score   support

Non-Conflicting       0.75      0.99      0.85      9751
    Conflicting       0.51      0.05      0.08      3287

    avg / total       0.69      0.75      0.66     13038
```

Gradient Boosting has an increased AUC compared to the Random Forest models but the recall has decreased quite a bit.

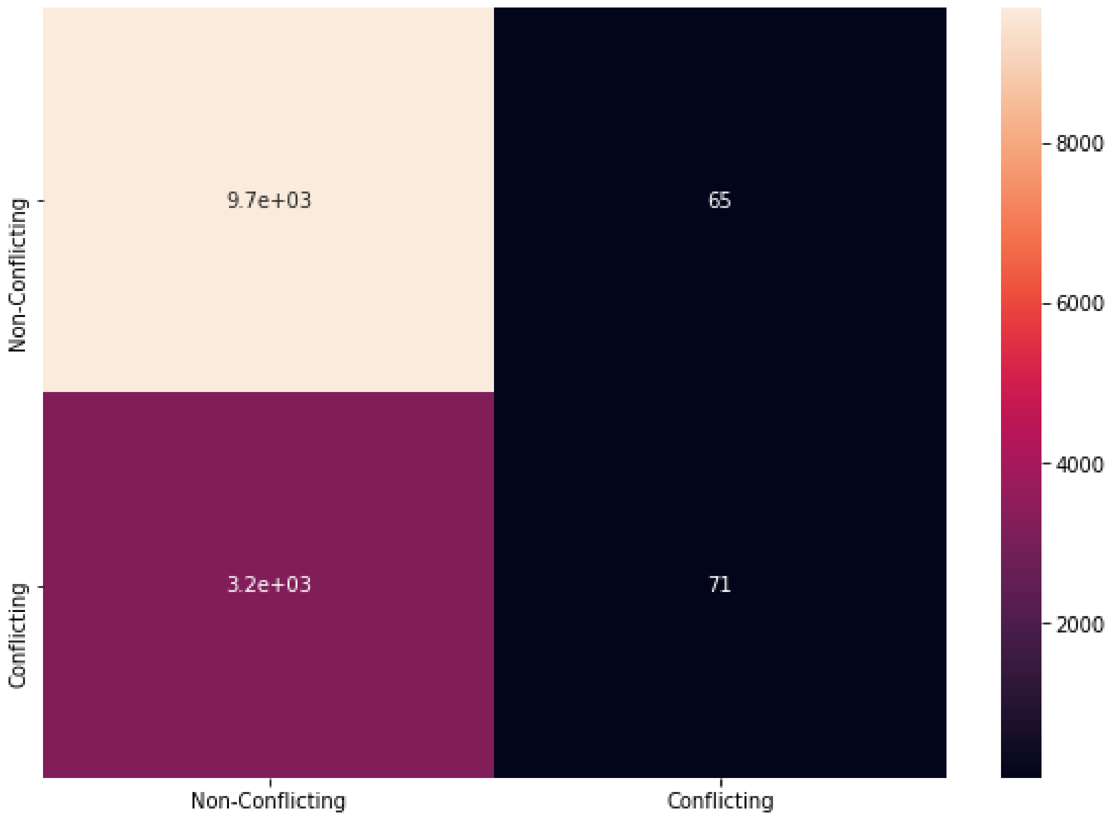
0.12 Gradient Boosting Grid Search

```
In [251]: gb_clf = ensemble.GradientBoostingClassifier()

          parameters = {'loss':['deviance', 'exponential'],
                        'min_samples_split':[2, 5, 10],
                        'max_depth':[5,6,7,8],
                        'max_features':['sqrt', 'log2'],
                        'n_estimators':[100,300,600]}

          #fitting model and printing best parameters and score from model
          grid_gb = GridSearchCV(gb_clf, param_grid=parameters, refit='recall_score', n_jobs=2)
          grid_gb.fit(X_train, y_train)

          # Train
          gb_preds_train = grid_gb.predict_proba(X_train)
          gb_probs_train = gb_preds_train[:, 1]
          gb_roc_train = roc_auc_score(y_train, gb_probs_train)
```

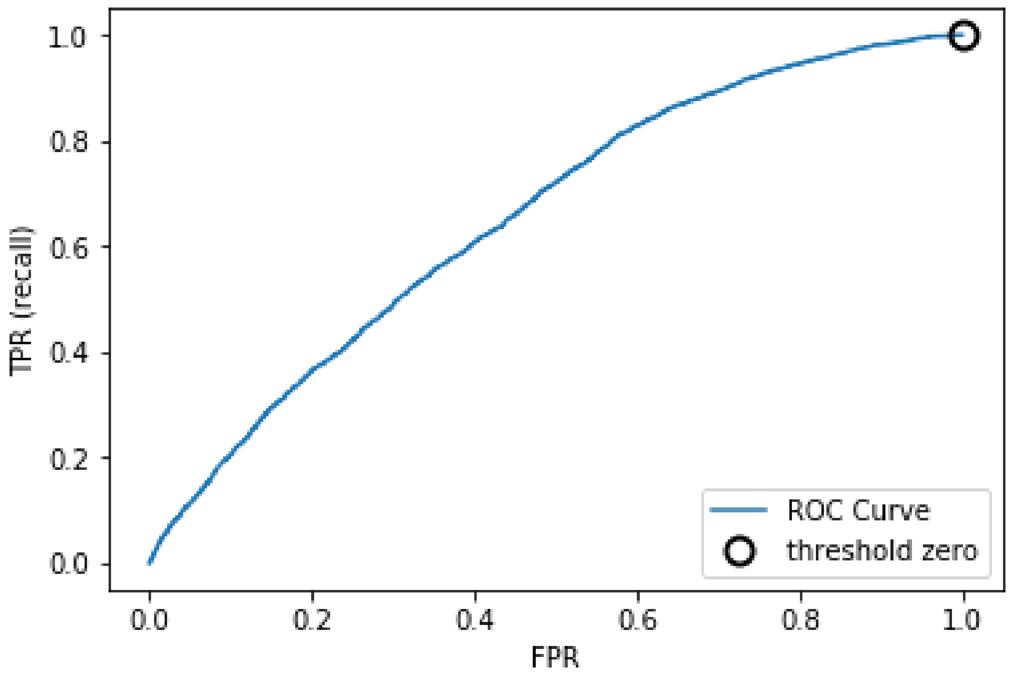
```
    # Dev
    gb_preds_dev = grid_gb.predict_proba(X_dev)
    gb_probs_dev = gb_preds_dev[:, 1]
    gb_roc_dev = roc_auc_score(y_dev, gb_probs_dev)

    # Confusion matrix
    conf_matrix_gb_dev = confusion_matrix(y_dev, np.argmax(gb_preds_dev, axis=1))

    # Best parameters from the grid search
    best_params = grid_gb.best_params_
    print(f'Best Parameters: {best_params}\n')

    print(f'roc_auc_score Train: {gb_roc_train}')
    print(f'roc_auc_score Dev: {gb_roc_dev}\n')
    print(f'Confusion Matrix: \n {conf_matrix_gb_dev} \n')
    print(classification_report(y_dev, np.argmax(gb_preds_dev, axis=1), target_names=['No

    # Classification report and heatmap
    df_cm = pd.DataFrame(conf_matrix_gb_dev, index = [i for i in ['Non-Conflicting', 'Con
                      columns = ['Non-Conflicting', 'Conflicting'])

    plt.figure(figsize = (10,7))
    sns.heatmap(df_cm, annot=True)
```

```
Best Parameters: {'loss': 'exponential', 'max_depth': 7, 'max_features': 'log2', 'min_samples_s

roc_auc_score Train: 0.7563508114006317
roc_auc_score Dev: 0.6559383408040619

Confusion Matrix:
 [[9686   65]
 [3216   71]]

                 precision    recall  f1-score   support

Non-Conflicting       0.75      0.99      0.86      9751
    Conflicting       0.52      0.02      0.04      3287

    avg / total       0.69      0.75      0.65     13038
```

```
Out[251]: <matplotlib.axes._subplots.AxesSubplot at 0x1a3d88d2b0>
```



```
In [252]: # Plot ROC curve
          plot_roc_curve(y_dev, gb_probs_dev)
```



Much like with the Random Forest model, the performance decreased after grid search.

0.13 Random Forest was the best performing model

```
In [259]: # Test - rfc is model above
          rfc_preds_test = rfc.predict_proba(X_test)
          rfc_probs_test = rfc_preds_test[:, 1]
          rfc_roc_test = roc_auc_score(y_test, rfc_probs_test)

          # AUC Scores
          print(f'roc_auc_score Test: {rfc_roc_test}\n')

          # Confusion Matrix
          conf_matrix_rf_test = confusion_matrix(y_test, np.argmax(rfc_preds_test, axis=1))
          print(f'Confusion Matrix:\n{conf_matrix_rf_test}\n')

          # Classification Report
          print(classification_report(y_test, np.argmax(rfc_preds_test, axis=1), target_names=
```

```
roc_auc_score Test: 0.62440682641834

Confusion Matrix:
[[8864  887]
 [2717  570]]

                 precision    recall  f1-score   support
```

```
Non-Conflicting       0.77      0.91      0.83      9751
    Conflicting       0.39      0.17      0.24      3287

    avg / total       0.67      0.72      0.68     13038
```

0.14 Conclusion

While achieving an AUC level of 62% using Random Forest is not immediately impressive, I think applying machine learning to the field of genomics is a step in the right direction. While I have no domain knowledge in this specific area, I imagine it would be beneficial to be able to predict at better than random, whether or not a genetic variation will result in conflicting classifications. Given this additional knowledge, labs or clinicians could add additional levels of inspection to overcome the increased likelihood of a conflicting classification.

0.15 Challenges I faced

The most challenging aspect of this problem was preprocessing the data and interpreting results on an imbalanced dataset.

- There were quite a few categorical variables that needed to be converted to a binary form.
- Some of the categorical variables also consisted of thousands of unique values.
- Every row consisted of NaN values, which meant imputing the data was a necessity.
- I had no domain knowledge in the area, so feature engineering was not an option, which likely hindered my ability to improve the accuracy of my models
- The dataset was imbalanced so using accuracy as a performance metric won't work. I had to examine the ROC, AUC, precision, recall and f1-score.