Importing Data/Packages

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree C
lassifier
from sklearn.model_selection import train_test_split # Import train_test
_split function
from sklearn import metrics #Import scikit-learn metrics module for accu
racy calculation
```

In [3]:
```
dataset= pd.read_csv('https://raw.githubusercontent.com/pleunipennings/E
DSS/df328799ceb8dc834c63669645f79ee44dbf1abc/allfeaturesV8_EDSS.csv')
dataset
```

Out[3]:

	pos	makesCpG	bigAAChange	Avg_Mutation_Freq	High_Mutation_Freq	5' UTR	Core	E1
0	8621	0	0	0.000009	0	0	0	0
1	8622	0	1	0.000019	0	0	0	0
2	8627	0	0	0.000019	0	0	0	0
3	8633	0	0	0.000024	0	0	0	0
4	8634	1	1	0.000024	0	0	0	0
...	...	...	...	...	...	...	...	...
7879	3191	0	0	0.299070	1	0	0	0
7880	3473	0	0	0.299204	1	0	0	0
7881	1512	0	1	0.299441	1	0	0	0
7882	6971	0	0	0.299442	1	0	0	0
7883	800	1	0	0.299490	1	0	1	0

7884 rows × 29 columns

In [4]:
```
dataset.describe() #provides dataframe overview
```

Out[4]:

	pos	makesCpG	bigAAChange	Avg_Mutation_Freq	High_Mutation_Freq	5' U
count	7884.000000	7884.000000	7884.000000	7884.000000	7884.000000	7884.00(
mean	4452.503171	0.146372	0.503425	0.031108	0.504693	0.00(
std	2427.644331	0.353502	0.500020	0.055110	0.500010	0.09(
min	264.000000	0.000000	0.000000	0.000009	0.000000	0.00(
25%	2346.750000	0.000000	0.000000	0.002362	0.000000	0.00(
50%	4460.500000	0.000000	1.000000	0.004376	1.000000	0.00(
75%	6552.250000	0.000000	1.000000	0.033633	1.000000	0.00(
max	8641.000000	1.000000	1.000000	0.299490	1.000000	1.00(

8 rows × 29 columns

In [5]:
```
dataset.dtypes #can see data classes for each parameter in data
```

Out[5]:
```
pos                     int64
makesCpG                int64
bigAAChange             int64
Avg_Mutation_Freq     float64
High_Mutation_Freq      int64
5' UTR                  int64
Core                    int64
E1                      int64
E2                      int64
HVR1                    int64
NS1                     int64
NS2                     int64
NS3                     int64
NS4A                    int64
NS4B                    int64
NS5A                    int64
NS5B                    int64
Mutation_Rate         float64
RNAstructure            int64
Nonsyn                  int64
Positive AA             int64
Negative AA             int64
Hydrophobic AA          int64
Polar AA                int64
Nonpolar AA             int64
Acidic AA               int64
Basic AA                int64
Stop                    int64
Drastic                 int64
dtype: object
```

Looking for any missing data

```
In [6]: (dataset=='?').sum() #no missing data found
```

```
Out[6]: pos                     0
        makesCpG                0
        bigAAChange             0
        Avg_Mutation_Freq       0
        High_Mutation_Freq      0
        5' UTR                  0
        Core                    0
        E1                      0
        E2                      0
        HVR1                    0
        NS1                     0
        NS2                     0
        NS3                     0
        NS4A                    0
        NS4B                    0
        NS5A                    0
        NS5B                    0
        Mutation_Rate           0
        RNAstructure            0
        Nonsyn                  0
        Positive AA             0
        Negative AA             0
        Hydrophobic AA          0
        Polar AA                0
        Nonpolar AA             0
        Acidic AA               0
        Basic AA                0
        Stop                    0
        Drastic                 0
        dtype: int64
```

Split target and features

In [7]:
```
features = dataset.drop(columns='High_Mutation_Freq') #isolate column of
data from dataframe to use for prediction/analysis
features2 = features.drop(columns="Avg_Mutation_Freq")
features2
```

Out[7]:

	pos	makesCpG	bigAAChange	5' UTR	Core	E1	E2	HVR1	NS1	NS2	...	Nonsyn	Posit
0	8621	0	0	0	0	0	0	0	0	0	...	0	
1	8622	0	1	0	0	0	0	0	0	0	...	1	
2	8627	0	0	0	0	0	0	0	0	0	...	0	
3	8633	0	0	0	0	0	0	0	0	0	...	0	
4	8634	1	1	0	0	0	0	0	0	0	...	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7879	3191	0	0	0	0	0	0	0	0	1	...	0	
7880	3473	0	0	0	0	0	0	0	0	0	...	0	
7881	1512	0	1	0	0	0	0	1	0	0	...	1	
7882	6971	0	0	0	0	0	0	0	0	0	...	0	
7883	800	1	0	0	1	0	0	0	0	0	...	0	

7884 rows × 27 columns

In [8]:
```
labels = dataset["High_Mutation_Freq"] #saves target columns into variab
le, "labels"
print(labels)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
7879    1
7880    1
7881    1
7882    1
7883    1
Name: High_Mutation_Freq, Length: 7884, dtype: int64
```

In [9]:
```
labels = np.where(labels >= 1,1,0) #convert labels to binary values that
show either high or low mutation frequency
labels
```

Out[9]:
```
array([0, 0, 0, ..., 1, 1, 1])
```

In [10]:
```
print("Number of sites with low mutation frequency = " + str(np.count_no
nzero(labels==0))) #organizing data into number of sites with high or lo
w mutation frequencies
print("Number of sites with high mutation frequency = " + str(np.count_n
onzero(labels==1)))
```

```
Number of sites with low mutation frequency = 3905
Number of sites with high mutation frequency = 3979
```

Train vs. Test Data

In [11]:
```
features2_train, features2_test, labels_train, labels_test = train_test_
split(features2, labels, test_size=0.3, random_state=42) #random_state=
 randomizing the data in a standard way for everyone
# 70% training and 30% test
```

In [12]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(max_depth = 4)
#max_depth: no more than x number of questions

# Train Decision Tree Classifer
clf = clf.fit(features2_train,labels_train )
# .fit function creates the decision tree
```

Predictions

In [13]:
```
#Predict the response for test dataset
labels_pred = clf.predict(features2_test)
labels_pred
```

Out[13]:
```
array([0, 1, 0, ..., 0, 1, 1])
```

In [14]:
```
# Look at the predicted values. Remember, 0 means no constricted vessel
s, 1 means at least one.
print(labels_pred)
# And the real values.
print(labels_test)
```

```
[0 1 0 ... 0 1 1]
[0 1 1 ... 0 1 0]
```

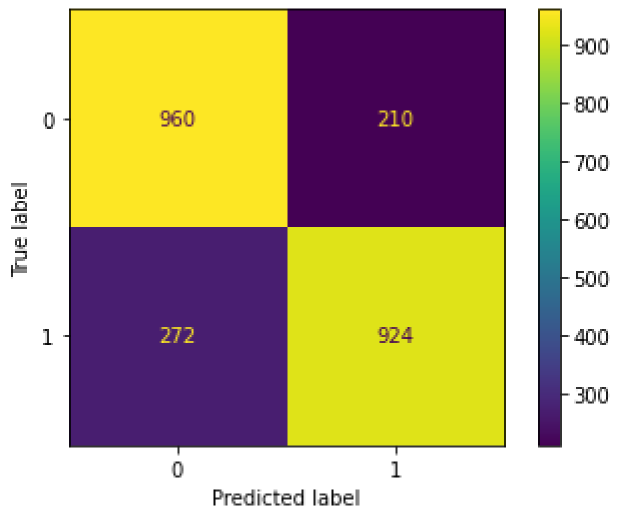
Results/Confusion Matrix

In [15]:
```
print(metrics.confusion_matrix(labels_test, labels_pred)) #There is 100%
in prediction of sites of high and low frequency mutations.
metrics.plot_confusion_matrix(clf, features2_test, labels_test)
```

```
[[960 210]
 [272 924]]

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function `
plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions o
r ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[15]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f2
444c74fd0>
```



In [16]:
```
acc = round(100 * metrics.accuracy_score(labels_test, labels_pred),2)
print("Accuracy:",acc,"%")
```

```
Accuracy: 79.63 %
```

Decision Tree

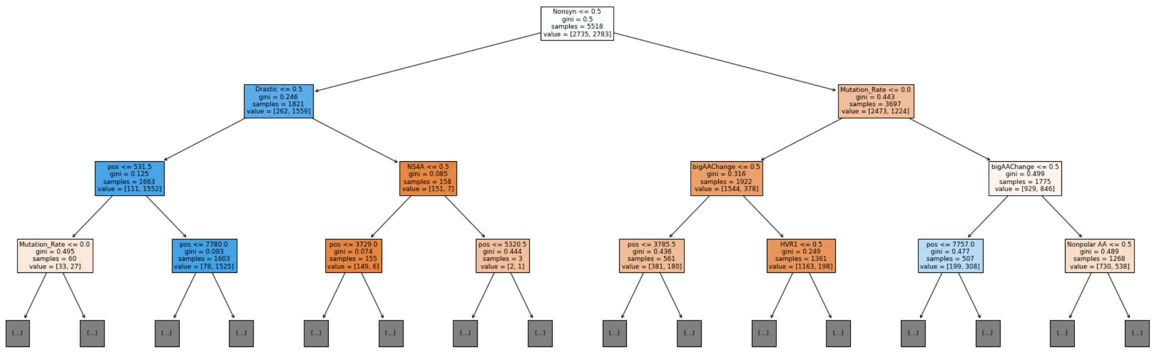
In [17]:
```
from matplotlib import pyplot as plt
from sklearn import tree
```

In [18]:
```
features2.columns[:-1]
```

Out[18]:
```
Index(['pos', 'makesCpG', 'bigAAChange', '5' UTR', 'Core', 'E1', 'E2',
'HVR1',
       'NS1', 'NS2', 'NS3', 'NS4A', 'NS4B', 'NS5A', 'NS5B', 'Mutation_R
ate',
       'RNAstructure', 'Nonsyn', 'Positive AA', 'Negative AA',
       'Hydrophobic AA', 'Polar AA', 'Nonpolar AA', 'Acidic AA', 'Basic
AA',
       'Stop'],
      dtype='object')
```

In [19]:
```
fig = plt.figure(figsize=(30,10)) #decision tree
tree.plot_tree(clf, filled=True, max_depth=3, feature_names = features2.
columns)
```

```
Out[19]: [Text(0.5, 0.9, 'Nonsyn <= 0.5\ngini = 0.5\nsamples = 5518\nvalue = [27
         35, 2783]'),
          Text(0.25, 0.7, 'Drastic <= 0.5\ngini = 0.246\nsamples = 1821\nvalue =
         [262, 1559]'),
          Text(0.125, 0.5, 'pos <= 531.5\ngini = 0.125\nsamples = 1663\nvalue =
         [111, 1552]'),
          Text(0.0625, 0.3, 'Mutation_Rate <= 0.0\ngini = 0.495\nsamples = 60\nv
         alue = [33, 27]'),
          Text(0.03125, 0.1, '\n  (...)  \n'),
          Text(0.09375, 0.1, '\n  (...)  \n'),
          Text(0.1875, 0.3, 'pos <= 7780.0\ngini = 0.093\nsamples = 1603\nvalue
         = [78, 1525]'),
          Text(0.15625, 0.1, '\n  (...)  \n'),
          Text(0.21875, 0.1, '\n  (...)  \n'),
          Text(0.375, 0.5, 'NS4A <= 0.5\ngini = 0.085\nsamples = 158\nvalue = [1
         51, 7]'),
          Text(0.3125, 0.3, 'pos <= 3729.0\ngini = 0.074\nsamples = 155\nvalue =
         [149, 6]'),
          Text(0.28125, 0.1, '\n  (...)  \n'),
          Text(0.34375, 0.1, '\n  (...)  \n'),
          Text(0.4375, 0.3, 'pos <= 5320.5\ngini = 0.444\nsamples = 3\nvalue =
         [2, 1]'),
          Text(0.40625, 0.1, '\n  (...)  \n'),
          Text(0.46875, 0.1, '\n  (...)  \n'),
          Text(0.75, 0.7, 'Mutation_Rate <= 0.0\ngini = 0.443\nsamples = 3697\nv
         alue = [2473, 1224]'),
          Text(0.625, 0.5, 'bigAAChange <= 0.5\ngini = 0.316\nsamples = 1922\nva
         lue = [1544, 378]'),
          Text(0.5625, 0.3, 'pos <= 3785.5\ngini = 0.436\nsamples = 561\nvalue =
         [381, 180]'),
          Text(0.53125, 0.1, '\n  (...)  \n'),
          Text(0.59375, 0.1, '\n  (...)  \n'),
          Text(0.6875, 0.3, 'HVR1 <= 0.5\ngini = 0.249\nsamples = 1361\nvalue =
         [1163, 198]'),
          Text(0.65625, 0.1, '\n  (...)  \n'),
          Text(0.71875, 0.1, '\n  (...)  \n'),
          Text(0.875, 0.5, 'bigAAChange <= 0.5\ngini = 0.499\nsamples = 1775\nva
         lue = [929, 846]'),
          Text(0.8125, 0.3, 'pos <= 7757.0\ngini = 0.477\nsamples = 507\nvalue =
         [199, 308]'),
          Text(0.78125, 0.1, '\n  (...)  \n'),
          Text(0.84375, 0.1, '\n  (...)  \n'),
          Text(0.9375, 0.3, 'Nonpolar AA <= 0.5\ngini = 0.489\nsamples = 1268\nv
         alue = [730, 538]'),
          Text(0.90625, 0.1, '\n  (...)  \n'),
          Text(0.96875, 0.1, '\n  (...)  \n')]
```



Lab Reflection

Questions

Describe what you did (describe the steps you took in the notebook) What you found (describe your conclusions from the decision tree; which features are most important in predicting the mutation frequency? what is the accuracy of your prediction?).

Write 300-400 words (as a team).

We took various steps, such as reading the Hep C data csv file into this notebook, checking for any gaps of missing data in the dataset, removing the target column of data from the dataset, etc. We also made sure to remove average mutation frequency, as this could hinder what/how predictions were made. We converted the column into binary data, and split it into 70% train and 30% test data. Then, we created a decision tree and confusion matrix. The confusion matrix and accuracy functions showed the model was 79.63% accurate at correctly predicting high and low frequency mutations.

The decision tree separated the data into synonymous vs. non-synonymous mutations. (Synonymous mutations result in no change to the amino acid being coded, whereas non-synonymous mutations results in a change to the amino acid and therefore the protein formed.) Then, on the right side, it separated into high vs. low frequency mutations. Low frequency mutations are more of a threat to the virus's survival, as opposed to high frequency mutations. On the right side, the it was further divided into how big of a change the mutations resulted in to affect the amino acid being coded.

The higher the certainty is about the predictions, the bolder the colors are. This can be seen towards the bottom of the decision tree. The gini index also gets lower towards the bottom of the tree to represent more unbalanced data. It also appears that the synonymous mutations were more accurately predicted as opposed to non-synonymous mutations, based on how bold the colors are in the decision tree.