In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [18]:
```
df = pd.read_csv("last_two_years_accidents.csv")
```

In [19]:
```
df.columns
```

Out[19]:
```
Index(['ID', 'Source', 'Severity', 'Start_Time', 'End_Time', 'Start_Lat',
       'Start_Lng', 'Distance(mi)', 'Street', 'City', 'County', 'State',
       'Zipcode', 'Country', 'Timezone', 'Airport_Code', 'Weather_Timestamp',
       'Temperature(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)',
       'Wind_Direction', 'Wind_Speed(mph)', 'Weather_Condition', 'Amenity',
       'Bump', 'Crossing', 'Give_Way', 'Junction', 'No_Exit', 'Railway',
       'Roundabout', 'Station', 'Stop', 'Traffic_Calming', 'Traffic_Signal',
       'Turning_Loop', 'Sunrise_Sunset', 'Civil_Twilight', 'Nautical_Twilight',
       'Astronomical_Twilight'],
      dtype='object')
```

In [20]:
```
severity_labels = {
    1: 'Not Severe',
    2: 'Not Severe',
    3: 'Severe',
    4: 'Severe'
}
```

In [21]:
```
df['Severity'] = df['Severity'].map(severity_labels)
```

In [22]:
```
size = df['Severity'].value_counts()['Severe']
```

In [23]:
```
df_balanced_severity = pd.DataFrame()
```

In [24]:
```
df_balanced_severity = df.groupby('Severity', group_keys = False).apply(lambda x: x.sample(size, random_state = 30))
```

In [25]:
```
df_balanced_severity['Severity'].value_counts()
```

Out[25]:
```
Severity
Not Severe    128292
Severe        128292
Name: count, dtype: int64
```

In [26]:
```
categorical_features = set(["Weather_Condition", "Civil_Twilight", 'Wind_Direction'])
```

In [27]:
```
for feature in categorical_features:
    df_balanced_severity[feature] = df_balanced_severity[feature].astype("category")
```

In [28]:
```
bool_columns = df_balanced_severity.select_dtypes(include='bool').columns
```

In [29]:
```
df_balanced_severity[bool_columns] = df_balanced_severity[bool_columns].replace({True:1, False:0})
```

In [30]:
```
df2= df_balanced_severity[['Start_Lat','Start_Lng','Distance(mi)', 'Temperature(F)', 'Humidity(%)', 'Pressure(in)',
           'Visibility(mi)', 'Wind_Speed(mph)','Amenity','Bump','Crossing','Give_Way',
           'Junction','No_Exit','Railway','Roundabout','Station','Stop','Traffic_Calming','Traffic_Signal',
           'Civil_Twilight','Weather_Condition','Civil_Twilight',
           'Wind_Direction','Severity']]
```

In [31]:
```
df2 = pd.get_dummies(df2, columns=list(categorical_features), drop_first=True)
```

In [32]:
```
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [33]:
```
Y = df2['Severity'] # target column
X = df2.drop(columns = ['Severity']) # features
```

In [34]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y,test_size=0.2, random_state=30)
```

In [35]:

```
DTree1 = DecisionTreeClassifier()
DTree1.fit(X_train, y_train)
y_test_pred = DTree1.predict(X_test)
y_train_pred = DTree1.predict(X_train)


acc = accuracy_score(y_test, y_test_pred)
CM = confusion_matrix(y_test,y_test_pred)
print(f"Accuracy of the DTree : {np.round(acc,3)*100}")
print(f"Confusion Matrix: \n {CM}")
```

```
Accuracy of the DTree : 77.9
Confusion Matrix:
 [[20010  5680]
 [ 5646 19981]]
```

In [36]:

```
DTree1.get_params()
```

Out[36]:

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'monotonic_cst': None,
 'random_state': None,
 'splitter': 'best'}
```

In [37]:

```
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

  Not Severe       0.98      1.00      0.99    102602
      Severe       1.00      0.98      0.99    102665

    accuracy                           0.99    205267
   macro avg       0.99      0.99      0.99    205267
weighted avg       0.99      0.99      0.99    205267
```

In [39]:

```
print(classification_report(y_test, y_test_pred))
```

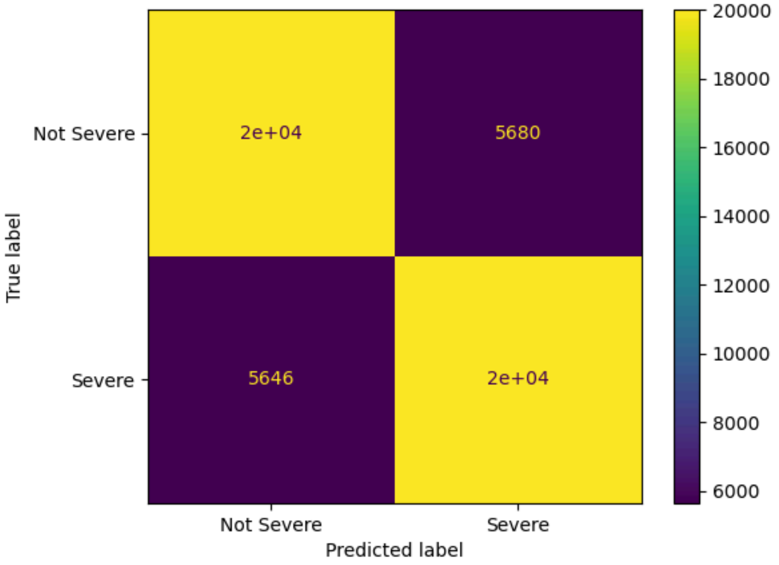
```
              precision    recall  f1-score   support

  Not Severe       0.78      0.78      0.78     25690
      Severe       0.78      0.78      0.78     25627

    accuracy                           0.78     51317
   macro avg       0.78      0.78      0.78     51317
weighted avg       0.78      0.78      0.78     51317
```

In [40]:

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_test_pred, labels=DTree1.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=DTree1.classes_)
disp.plot()
plt.show()
```



In [42]:

```
importances = pd.DataFrame(np.zeros((X_train.shape[1], 1)), columns=["importance"], index=X_train.columns)
```

In [43]:

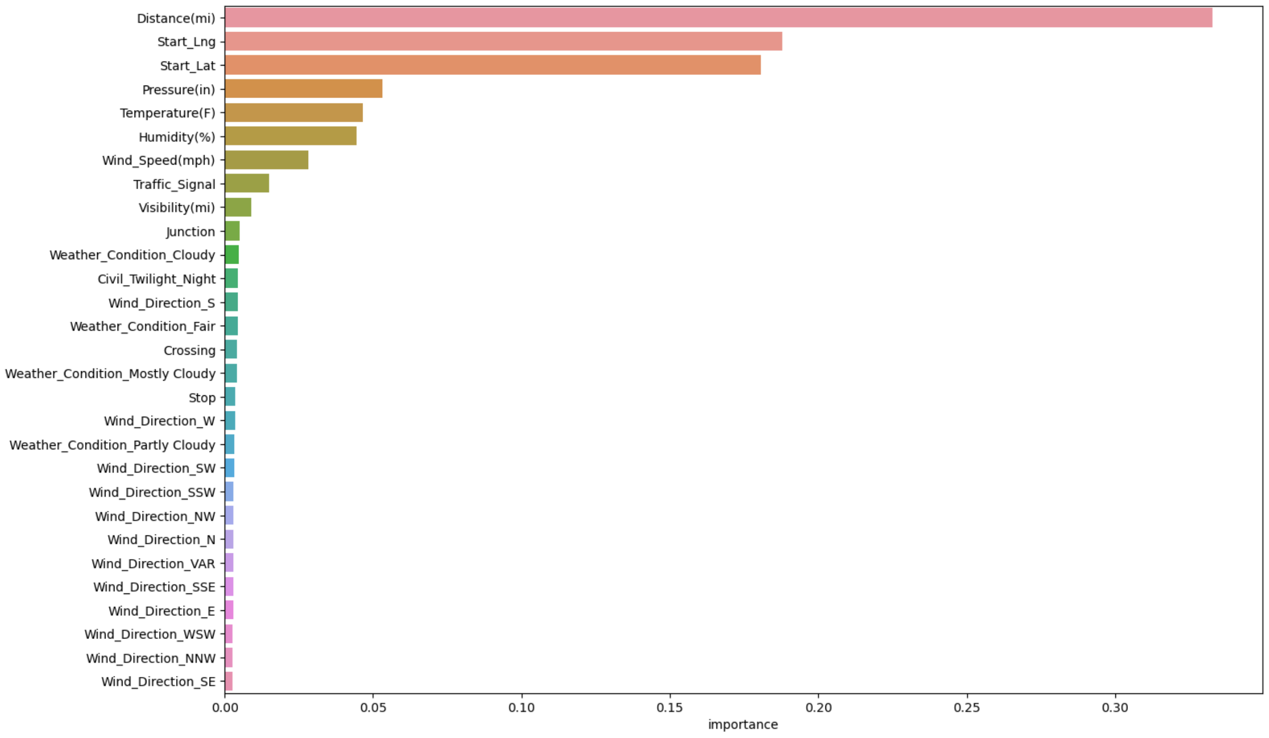
```
importances.iloc[:,0] = DTree1.feature_importances_
```

In [44]:

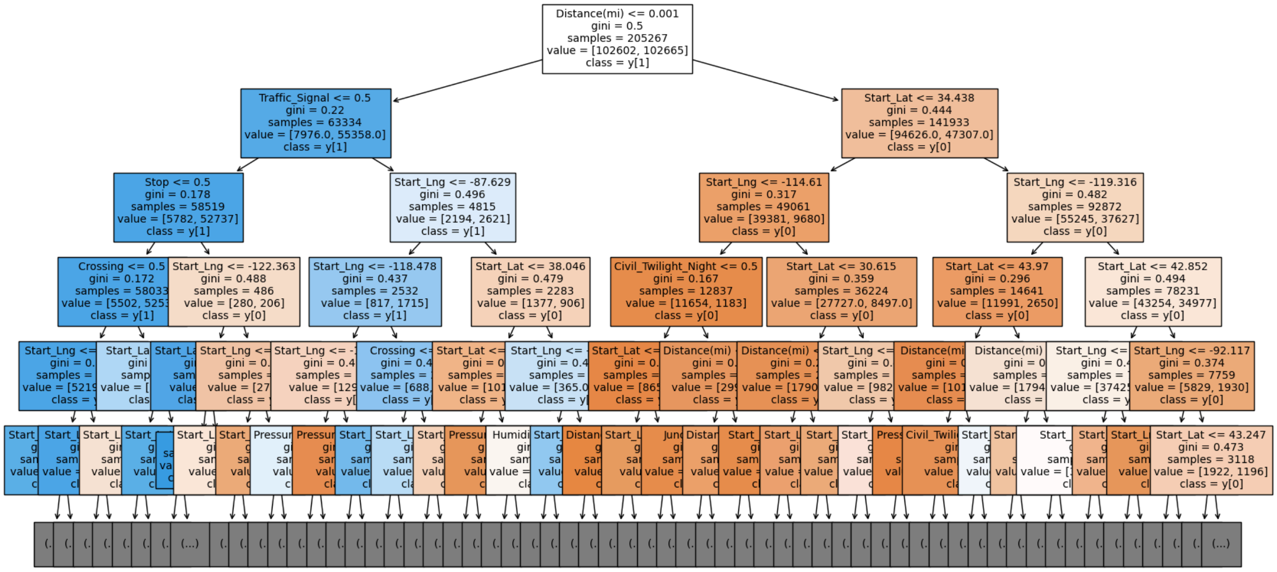
```
importances = importances.sort_values(by="importance", ascending=False)[:30]
```

In [45]:

```
plt.figure(figsize=(15, 10))
sns.barplot(x="importance", y=importances.index, data=importances)
plt.show()
```



In [50]:
```python
fig, ax = plt.subplots(figsize=(20, 10))
plot_tree(DTree1, max_depth=5, fontsize=10, feature_names=X_train.columns.to_list(), class_names = True, filled=True)
plt.show()
```



In [51]:
```python
dtc1 = DecisionTreeClassifier(random_state=42, criterion = 'entropy', max_depth = 5, min_samples_leaf=10000, min_samples_spli
```

In [52]:
```python
print("Default scores:")
dtc1.fit(X_train, y_train)
print("Train score:", dtc1.score(X_train, y_train))
print("Validation score:", dtc1.score(X_test, y_test))
```

```
Default scores:
Train score: 0.7306776052653373
Validation score: 0.732778611376347
```

In [53]:
```python
y_pred = dtc1.predict(X_test)
y_pred_train=dtc1.predict(X_train)
```

In [54]:
```python
accuracy_train= accuracy_score(y_train, y_pred_train)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
print("Accuracy_Train: {:.2f}%".format(accuracy_train * 100))
```

```
Accuracy: 73.28%
Accuracy_Train: 73.07%
```

In [55]:
```python
y_pred = dtc1.predict(X_test)
```

In [56]:
```python
accuracy_score(y_test, y_pred)
```

Out[56]:
```
0.732778611376347
```

In [58]:
```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

  Not Severe       0.67      0.93      0.78     25690
      Severe       0.88      0.54      0.67     25627

    accuracy                           0.73     51317
   macro avg       0.77      0.73      0.72     51317
weighted avg       0.77      0.73      0.72     51317
```

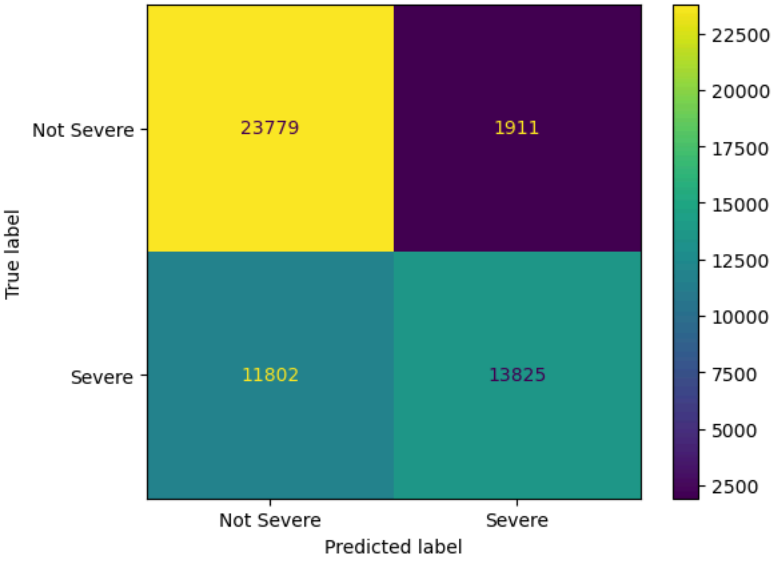
In [59]:
```python
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
```

In [60]:
```python
confmat
```

Out[60]:
```
array([[23779,  1911],
       [11802, 13825]], dtype=int64)
```

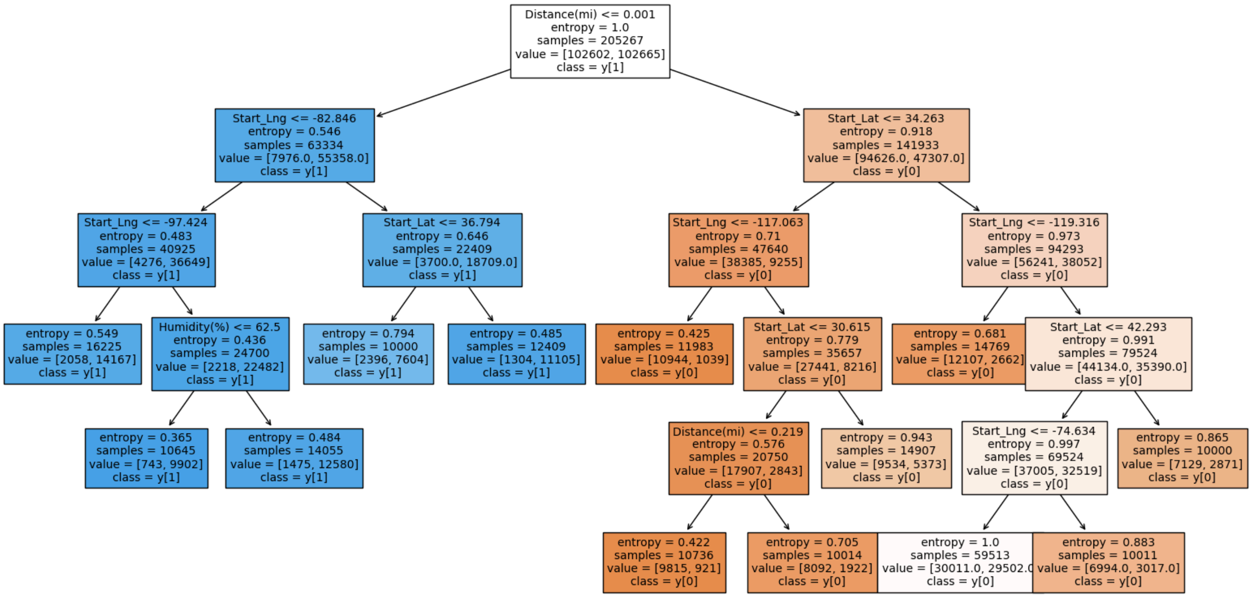
In [61]:

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_test_pred, labels=dtc1.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=confmat,
                              display_labels=dtc1.classes_)
disp.plot()
plt.show()
```



In [62]:

```
fig, ax = plt.subplots(figsize=(20, 10))
plot_tree(dtc1, max_depth=5, fontsize=10, feature_names=X_train.columns.to_list(), class_names = True, filled=True)
plt.show()
```



In [63]:

```
df3= df2.drop('Distance(mi)',axis = 1)
```

In [64]:

```
Y = df3['Severity'] # target column
X = df3.drop(columns = ['Severity']) # features
```

In [65]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y,test_size=0.2, random_state=30)
```

In [67]:

```
dtc2 = DecisionTreeClassifier(random_state=42, criterion = 'gini', max_depth = 5,  min_samples_leaf=500, min_samples_split= 1
```

In [68]:

```
print("Default scores:")
dtc2.fit(X_train, y_train)
print("Train score:", dtc2.score(X_train, y_train))
print("Validation score:", dtc2.score(X_test, y_test))
```

```
Default scores:
Train score: 0.6305056341253099
Validation score: 0.6334937739930238
```

In [69]:

```
y_pred = dtc2.predict(X_test)

accuracy_score(y_test, y_pred)
```

Out[69]:

```
0.6334937739930238
```

In [70]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

  Not Severe       0.68      0.50      0.58     25690
      Severe       0.61      0.76      0.68     25627

    accuracy                           0.63     51317
   macro avg       0.64      0.63      0.63     51317
weighted avg       0.64      0.63      0.63     51317
```

In [71]:

```
y_pred = dtc2.predict(X_test)
confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
```

In [72]:

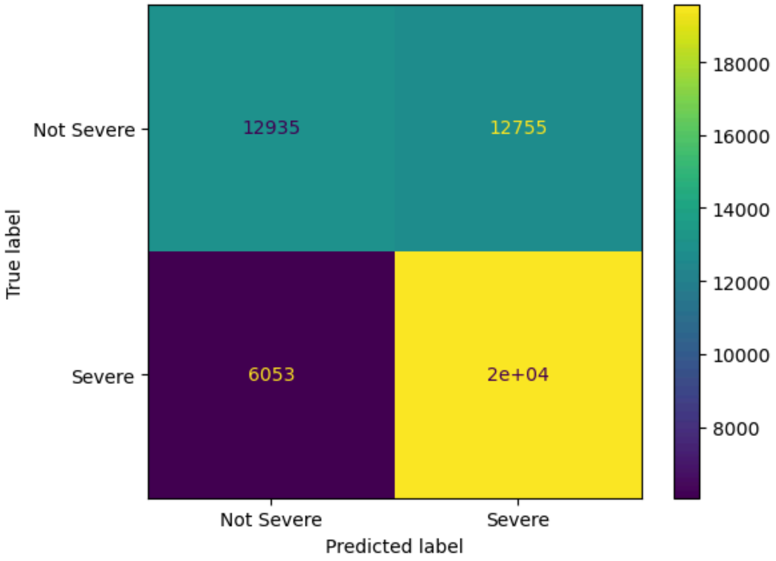
```
confmat
```

Out[72]:

```
array([[12935, 12755],
       [ 6053, 19574]], dtype=int64)
```

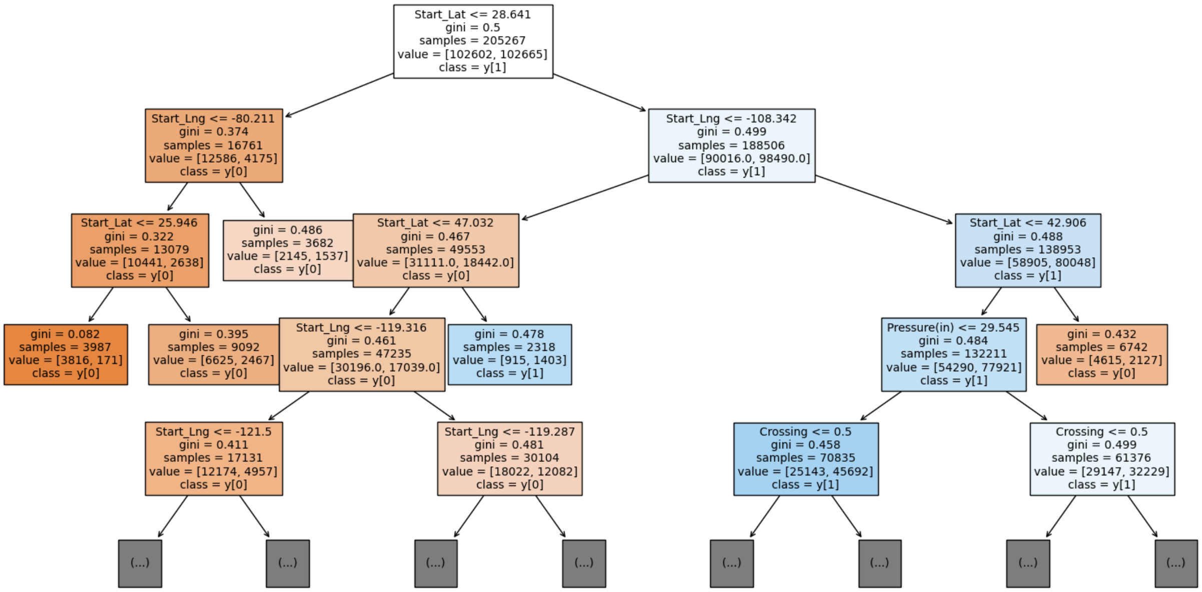
In [73]:

```
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_test_pred, labels=dtc2.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=confmat,
                              display_labels=dtc2.classes_)
disp.plot()
plt.show()
```



In [74]:

```
fig, ax = plt.subplots(figsize=(20, 10))
plot_tree(dtc2, max_depth=4, fontsize=10, feature_names=X_train.columns.to_list(), class_names = True, filled=True)
plt.show()
```



In [ ]: