Portfolio assignment 16

30 min: Train a decision tree to predict one of the categorical columns of your own dataset.

- Split your dataset into a train (70%) and test (30%) set.
- Use the train set to fit a DecisionTreeClassifier. You are free to to choose which columns you want to use as feature variables and you are also free to choose the max_depth of the tree.
- Use your decision tree model to make predictions for both the train and test set.
- Calculate the accuracy for both the train set predictions and test set predictions.
- Is the accurracy different? Did you expect this difference?
- Use the plot_tree function above to create a plot of the decision tree. Take a few minutes to analyse the decision tree. Do you understand the tree?

In [1]:
```
import pandas as pd
import seaborn as sns
```

In [2]:
```
steam = pd.read_csv('steam_games.csv')
steam.head()
```

Out[2]:

	url	types	name	desc_snippet	recent_revi
0	https://store.steampowered.com/app/379720/DOOM/	app	DOOM	Now includes all three premium DLC packs (Unto...	Very Posit (554),- 89% the 554 u re
1	https://store.steampowered.com/app/578080/PLAY...	app	PLAYERUNKNOWN'S BATTLEGROUNDS	PLAYERUNKNOWN'S BATTLEGROUNDS is a battle roya...	Mixed,(6,21 49% of 6,214 u review
2	https://store.steampowered.com/app/637090/BATT...	app	BATTLETECH	Take command of your own mercenary outfit of '...	Mixed,(16 54% of the user review
3	https://store.steampowered.com/app/221100/DayZ/	app	DayZ	The post-soviet country of Chernarus is struck...	Mixed,(93 57% of the user review
4	https://store.steampowered.com/app/8500/EVE_On...	app	EVE Online	EVE Online is a community-driven spaceship MMO...	Mixed,(28 54% of the user review

In [3]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [4]:
```
len(steam)
```

Out[4]:
```
40833
```

In [5]:
```
len(steam.dropna())
```

Out[5]:
```
82
```

that's a lot less rows... let's fill up the empty spaces.
we'll start by filling up the numerical values

In [6]:
```
steam = steam.fillna(value={'achievements': 0, 'discount_price': steam.original_price
```

In [7]:
```
len(steam.dropna(subset=['developer']))
```

Out[7]:
```
40490
```

This is still a fine size, plus it's weird to have games without a developer.

In [8]:
```
steam = steam.dropna(subset=['developer'])
```

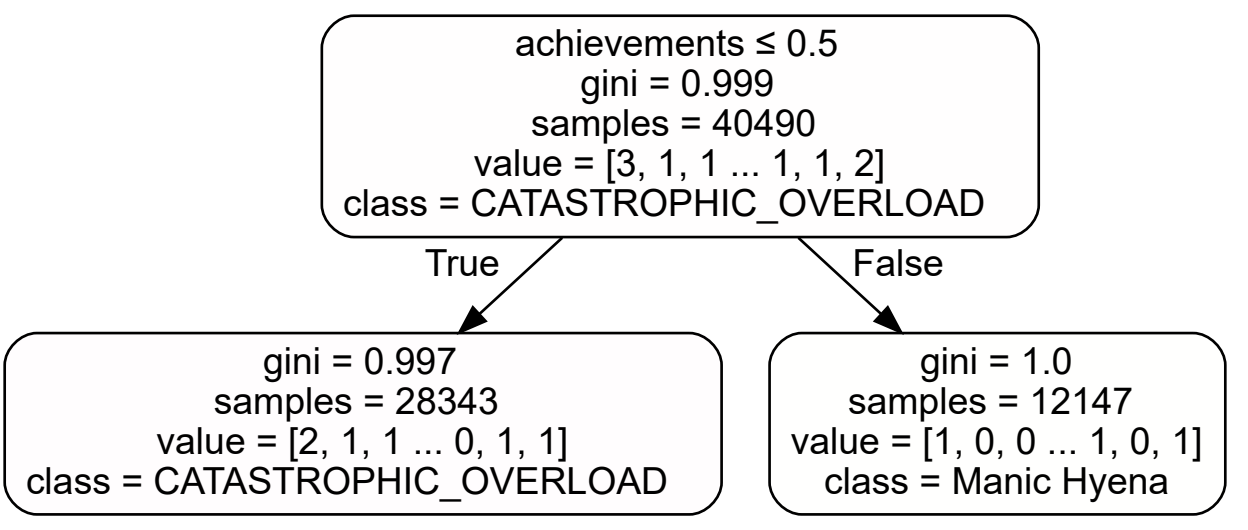
In [9]:
```
features= ['achievements']
dt = DecisionTreeClassifier(max_depth = 1) # Increase max_depth to see effect in the p
dt.fit(steam[features], steam['developer'])
```

Out[9]:
```
DecisionTreeClassifier(max_depth=1)
```

In [10]:
```
from sklearn import tree
import graphviz

def plot_tree_classification(model, features, class_names):
    # Generate plot data
    dot_data = tree.export_graphviz(model, out_file=None,
                                    feature_names=features,
                                    class_names=class_names,
                                    filled=True, rounded=True,
                                    special_characters=True)

    # Turn into graph using graphviz
    graph = graphviz.Source(dot_data)

    # Write out a pdf
    graph.render("decision_tree")

    # Display in the notebook
    return graph
```

In [11]:
```
plot_tree_classification(dt, features, steam.developer.unique())
```

Out[11]:



Here it asks if there are less than 0.5 achievements, it'll be from Tokiwa Graphics.

In [12]:
```
predictions = dt.predict(steam[features])
```

In [13]:
```
def calculate_accuracy(predictions, actuals):
    if(len(predictions) != len(actuals)):
        raise Exception("The amount of predictions did not equal the amount of actuals

    return (predictions == actuals).sum() / len(actuals)
```

In [14]:
```
calculate_accuracy(predictions, steam.developer)
```

Out[14]:
```
0.027587058532971102
```

I guess this has a very low accuracy.

In [15]:
```
len(steam.developer.unique())
```

Out[15]:
```
17420
```

This might be why

In [16]:
```
top10 = steam.developer.value_counts().sort_values(ascending=False).index[:10]
```

In [17]:
```
steam.loc[~steam.developer.isin(top10),'developer'] = 'Other'
```

In [18]:
```
len(steam.developer.unique())
```

Out[18]:
```
11
```

In [19]:
```
steam.developer.value_counts()
```

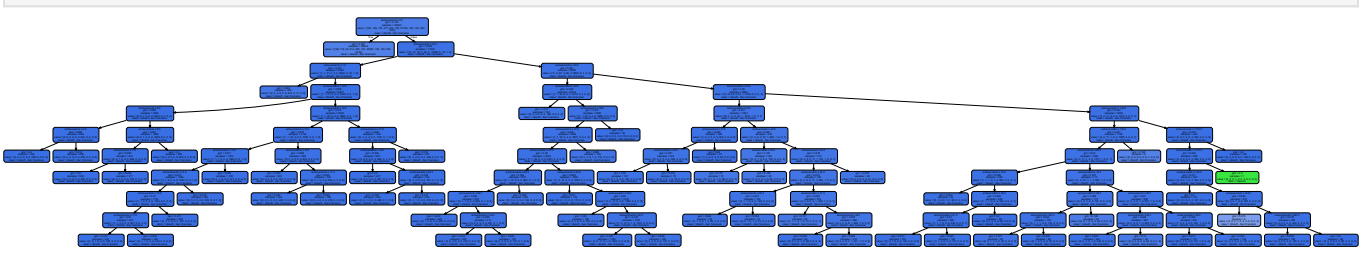
Out[19]:
```
Other                          37190
Ubisoft - San Francisco         1041
SmiteWorks USA, LLC              784
KOEI TECMO GAMES CO., LTD.       472
Dovetail Games                   217
Sly                              153
Paradox Development Studio       140
Capcom                           129
N3V Games                        126
CAPCOM Co., Ltd.                 120
Choice of Games                  118
Name: developer, dtype: int64
```

In [20]:
```
features= ['achievements']
dt = DecisionTreeClassifier(max_depth = 10) # Increase max_depth to see effect in the
dt.fit(steam[features], steam['developer'])
```

Out[20]:
```
DecisionTreeClassifier(max_depth=10)
```

In [21]:
```
plot_tree_classification(dt, features, steam.developer.unique())
```

Out[21]:



Apparently now the majority is Ubisoft, a fix would be to check per class.