



## PROJECT

## Titanic Survival Exploration

A part of the Machine Learning Engineer Nanodegree Program

## PROJECT REVIEW

## NOTES


SHARE YOUR ACCOMPLISHMENT!  

## Meets Specifications

Dear Student,

Great work with your project!. As a side comment in this [Kaggle section](#) there are other examples of Titanic that you can use to keep learning Pandas, matplotlib or Scikit-Learn.PD: I think [this is an excellent tutorial](#) to learn more on notebooks and this [tutorial for Markdown syntax](#). Note notebooks are a great tool and Markdown notation is becoming more and more popular, so it is definitely worth to invest some time to learn more about it.Congratulations on passing your exam and stay  


## Answers to Each Question

The `predictions_0` function has been run and the accuracy of the predictions is reported.Nice work! At this moment we have a pretty simple algorithm that always predicts that passengers did not survive. The `predictions_1` function has been correctly implemented. The expected accuracy of the predictions is reported.Well done implementing the new condition into your function. Now, using `Sex` to determine whether a passenger survived or not seems to be quite important since accuracy is significantly higher!Our model now assumes all females survived while all males perished and the accuracy boosted up to 78%, wow! this means that out of 100 predictions, our model is right in 78 of them 

So, had females higher survival rates than males? It seems so, actually, it seems that most females survived while most males perished!

The `predictions_2` function has been correctly implemented. The expected accuracy of the predictions is reported.Well done completing this predictor!. It seems `Age` is also important since it helps to improve our model a bit more. `males` with `age < 10` seems to mostly survive, it seems male Childs had higher survival rates than adults!.The `predictions_3` function has been correctly implemented and obtains a prediction accuracy of at least 80%. The approach to the task has been documented, including features that were explored and intermediate steps taken to complete the function.

Good dataset exploration and testing of the different features to identify those combinations with higher survival rates until reach an accuracy over 80%.

As an extra tip, although an accuracy between 80-83% is the best you can get, note you can deploy and visualize a [Decision Tree](#) with [Graphviz](#) to investigate how it splits the dataset into different features to maximize score (don't feel overwhelmed if you don't understand parts of this code, at this stage it is not expected you understand most of it, this is part of what you will learn in the NanoDegree! 

For example, I tried with `max_features=3` and `max_depth= 2` but feel free to change [parameters](#) to investigate different trees:

NOTE: Apart of having Graphviz already installed in your computer, you might need to install some of the libraries here used to draw the decision tree!

```
# Import required libraries: (some of them are not available in Anaconda!)
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn import tree
from sklearn.metrics import accuracy_score
from IPython.display import Image
import pydotplus
import pandas

# Load Data
data = pandas.read_csv('titanic_data.csv', sep=',')

# Define outcome, drop non used features and generate a binary variable for Sex:
outcomes = data['Survived']
data = data.drop(['Survived', 'Name', 'Ticket', 'Cabin', 'Embarked', 'PassengerId'], axis=1)
data.loc[:, 'Sex'] = data['Sex'].apply(lambda x: 1. if x == 'female' else 0.)

# impute missing values in the age column using Sex and Pclass:
data.loc[:, 'Age'] = data.groupby(['Sex', 'Pclass']).transform(lambda x: x.fillna(x.median()))

# Split the data into train/test sets: (train data is used to make the model learn from data and test data is used to estimate how well
# model generalized)
sss = StratifiedShuffleSplit(10, test_size=0.3, random_state=450)
for train_index, test_index in sss.split(data, outcomes):
    X_train = data.iloc[train_index]
    y_train = outcomes.iloc[train_index]
    X_test = data.iloc[test_index]
    y_test = outcomes.iloc[test_index]

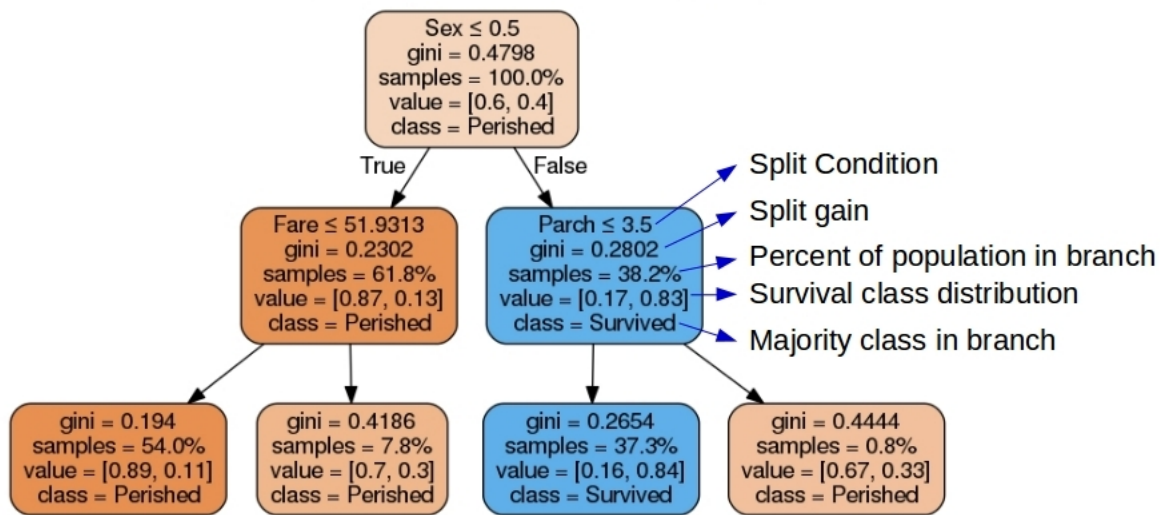
# Define Decision Tree to use: (give it a try and change parameters to see the different trees built)
clf = tree.DecisionTreeClassifier(max_features=3, max_depth=2)

# Use train data to train the model:
clf = clf.fit(X_train, y_train)

# Generate predictions over test set:
predictions = clf.predict(X_test)

# Accuracy results over test set:
print "Accuracy Score over test set:", accuracy_score(y_test, predictions)

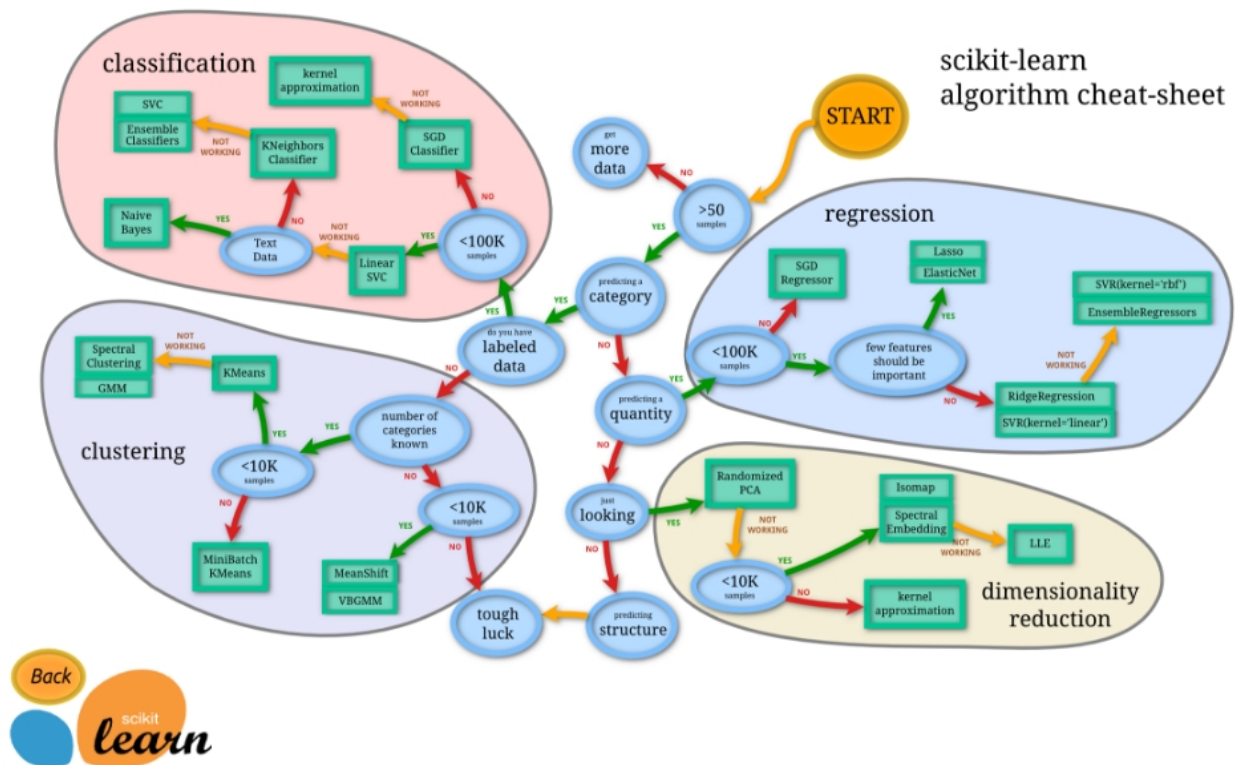
# Represent Generated Tree:
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=data.columns,
                                class_names=['Perished', 'Survived'],
                                filled=True, rounded=True,
                                proportion=True,
                                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



A valid scenario where supervised learning can be applied is reported. A clear outcome variable and at least two potential predictor variables are identified as part of the description.

That is exactly right, excellent example! 🍌. Check [Scikit implemented supervised algorithms](#) for your reference, you will use several of them along the NanoDegree.

Also, have a look at the [Scikit algorithm-cheat sheet](#) to help you find the right estimator for the job.



[DOWNLOAD PROJECT](#)

[RETURN TO PATH](#)

Rate this review