**Assignment Summary**

**Feature Extraction:**

When we think of a variety of a wine, our initial thoughts would be :

* Where is this wine made?
* which winery is it from?
* is it a costly wine?

These are generally the thoughts whenever we casually judge a variety of a wine.

Since we already have a wine dataset, why not search for answers by checking these columns:

* where is this wine made? (Country)
* which winery is it from? (Winery)
* is it a costly wine? (Price)

Can a computer model predict wine variety by using these features just like we do in real life?

The answer is YES.

So firstly, as per our initial analysis, the features selected were: Country, Winery and Price

Next step is to check if these features are correlated to wine variety by using mathematical calculations

**Step 1:**

However, to confirm whether or not they exhibit actual correlation to the target variable,

methods such as pps score and pandas profile were used to check the correlation.

A final list of features were deduced on the basis of pandas profile & pps score, as shown below:

|  |  |
| --- | --- |
| **Features** | **Predictive Power (pps) Score** |
| region\_1(The wine-growing area in a province or state ) | 0.4778 |
| province(The province or state that the wine is from) | 0.3658 |
| winery (The winery that made the wine) | 0.2090 |
| country(The country that the wine is from) | 0.1645 |
| price (The cost for a bottle of the wine) | 0.0783 |

**Step 2:**

All the top 5 features with high pps score & pandas profile report were selected (except “designation” attribute… since the “designation” attribute could not be used as it had more than 26k unique values which made it a limitation, computationally, on my PC.)

Hence this was replaced with the price column (since price has a logical relation to wine variety)

**Models Used:**

Since majority of the dataset consists of categorical data, classification algorithms were used to strengthen the predictive power of the model.

Below two models were used on the training data:

* Decision Tree Classifier
* Random Forest Classifier

Libraries used: numpy, sklearn, pandas, matplotlib, seaborn, ppscore

To determine the best model to use for this classification problem, a comparison was done between both the models.

The final result - **Random Forest classifier** performed slightly better than the **Decision Tree Classifier** in terms of accuracy.

The objective was to measure the performance of both the models on the training data.

Based on the results, **Random Forest model** was selected to be used on the test data

**Model Accuracy in train:**

|  |  |
| --- | --- |
| **Model Name** | **Accuracy on Training Data** |
| Decision Tree Classifier | 0.62711 |
| Random Forest Classifier | 0.64405 |

**Visualization with actionable insights**

This information is available in

Github folder - ‘ Data/Visualizations.ipnyb ’