**Meets Specifications**

**FEW LEARNING RESOURCES**

1. [Recursive Feature Selection](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html)
2. [Why AdaBoost works with Decision Tree](https://stats.stackexchange.com/questions/124628/why-adaboost-with-decision-trees)
3. [Why is logistic regression a linear classifier?](https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier)

**OVERALL COMMENTS**

Congratulations on finishing the project 🎉

This was a brilliant submission. The work was exceptional!

You did a great job and should be proud of yourself.

After reviewing this submission, I am impressed and satisfied with the effort and understanding put in to make this project a success.

All the requirements have been met successfully%

I have tried to provide you a detailed review by adding :-

* Few Suggestions which you can try and improve your model.
* Appreciation where you did great
* Some learning opportunities for improving your understanding beyond coursework

I hope you find the complete review informative  

Keep doing the great work and all the best for future project.

**Don't forget to rate my work as project reviewer! Your positive feedback is very helpful and appreciated - thank you!**

**Exploring the Data**

**Student's implementation correctly calculates the following:**

* **Number of records**
* **Number of individuals with income >$50,000**
* **Number of individuals with income <=$50,000**
* **Percentage of individuals with income > $50,000**

Well done on the correct Implementation. You could use an optimized version of data aggregation by using panda’s slicing technique

n\_greater\_50k = len(data[data.income == '>50K'])

EDA or exploratory data analysis are generally believed to be great technique when handling with data for machine learning. It helps in understanding the general nature of data in hand.  
You could check out the library [Seaborn](http://seaborn.pydata.org/examples/) which will give you an edge in EDA.  
For example

import seaborn as sns

sns.factorplot('income', 'capital-gain', hue='sex', data=data, kind='bar')

**Preparing the Data**

**Student correctly implements one-hot encoding for the feature and income data.**

Well done correct implementation. Correct number of one-hot encoded features identified  
There are a couple other ways that we can encode the labels here.  
One way is to use boolean indexing.

income = (income\_raw == ">50K").astype(np.uint8)

We can also use the LabelEncoder class provided by sklearn. This class is especially useful when we have lots of possible output labels. We can use it for this problem as follows:

encoder = LabelEncoder()

income = encoder.fit\_transform(income\_raw)

**Evaluating Model Performance**

**Student correctly calculates the benchmark score of the naive predictor for both accuracy and F1 scores.**

Great work on correctly calculating the Accuracy, Precision and Recall!  
Also, well done getting the right calculation of F-score.

You correctly calculated both accuracy and f-score. Good work!  
You could check this [link](https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall) for further understanding precision and recall.

[Beyond the F-1 score: A look at the F-beta score](https://medium.com/@douglaspsteen/beyond-the-f-1-score-a-look-at-the-f-beta-score-3743ac2ef6e3)

**The pros and cons or application for each model is provided with reasonable justification why each model was chosen to be explored.**

**Please list all the references you use while listing out your pros and cons.**

Very nice job mentioning some real-world application, strengths / weakness and reasoning for your choice! Great to know even outside of this project!  
Below you can find some pros and cons of industry methods

**Decision Tree**

* Typically very fast!
* Can handle both categorical and numerical features
* As we can definitely see here that our Decision Tree has an overfitting issue. This is typical with Decision Trees and Random Forests.
* They are are easy to visualize. Very interpretable model. Check out export\_graphviz

**Logistic Regression**

* The big thing that should be noted here is that a Logistic Regression model is a linear classifier. It cannot fit non-linear data. Thus, the model creates a single straight line boundary between the classes.  
  [How can we account for non-linear variable in logistic regression](http://stats.stackexchange.com/questions/79259/how-can-i-account-for-a-nonlinear-variable-in-alogistic-regression)  
  [Why is logistic regression a linear classifier](http://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier)
* Interpretable with some help
* Great for probabilities, since this works based on the sigmoid function
* Can set a threshold value!!

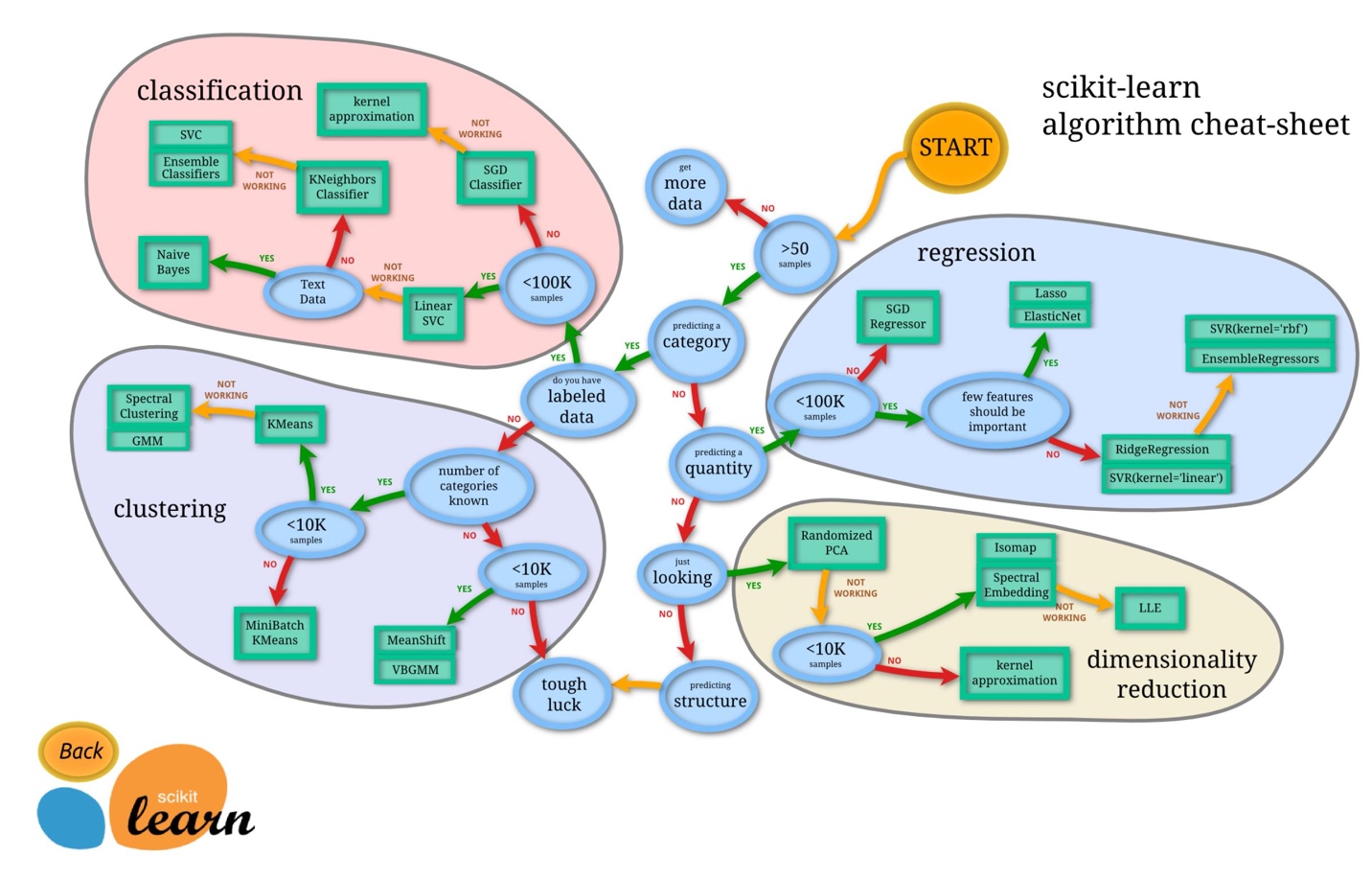
**Random Forest**

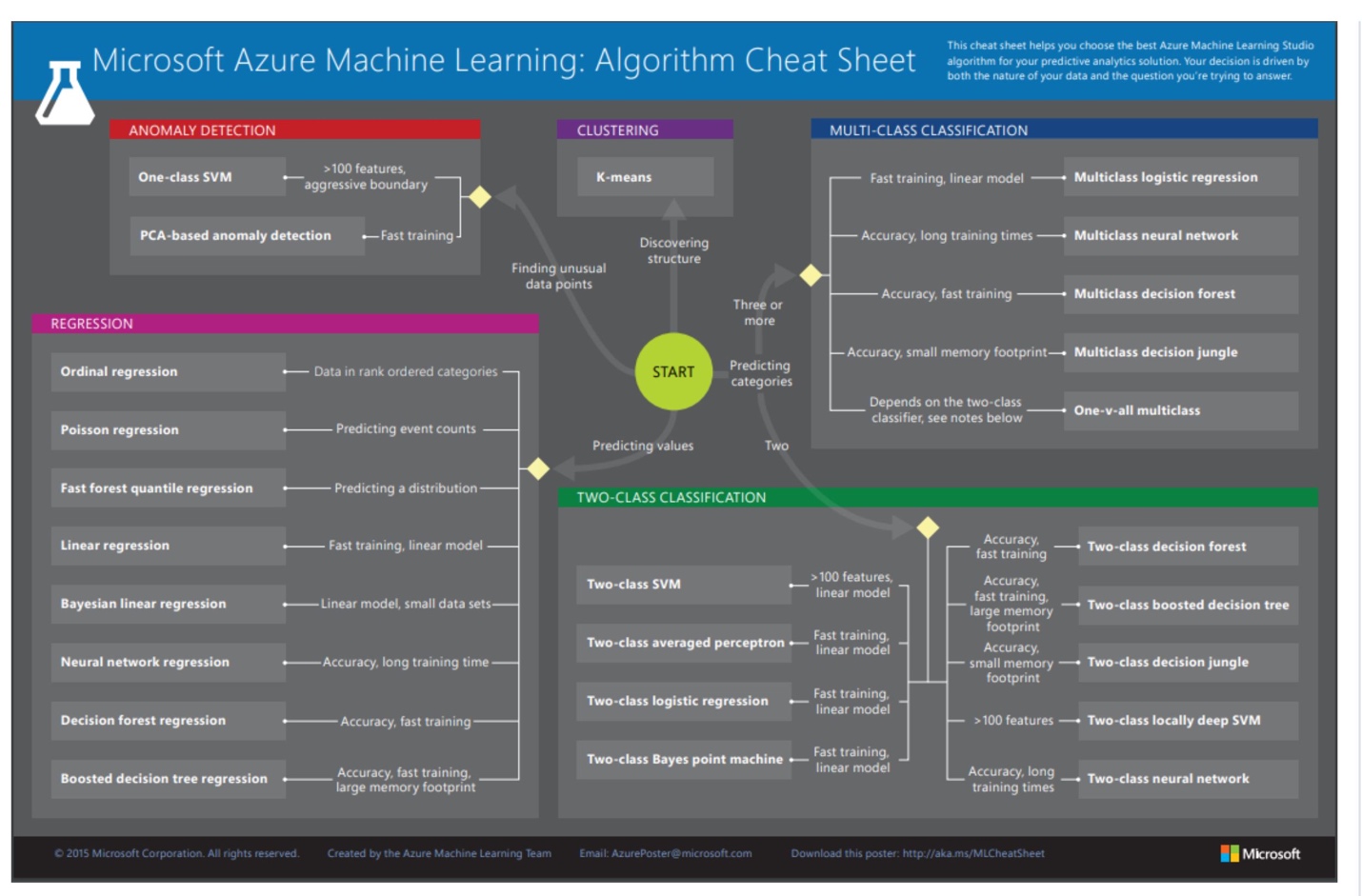
* Combines multiple decision trees which can eventually lead to a more robust model, typically reduce the variance.
* Can handle both categorical and numerical features
* Another great thing that a Random Forest model and tree methods in sklearn gives us is [feature importances](https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html). Which we use later on.

**SVM**

* Typically much slower, but the kernel trick makes it very nice to fit non-linear data.
* They are memory efficient, as they only have to remember the support vectors.
* Also note that the SVM output parameters are not really interpretable.
* We can use probability = True to calculate probabilities, however very slow as it used five fold CV

You can also check out this flowchart from Sklearn as a rough guideline(take it with a grain of salt however)

[](https://udacity-reviews-uploads.s3.us-west-2.amazonaws.com/_attachments/538361/1588286318/sklearn.JPG)

[](https://udacity-reviews-uploads.s3.us-west-2.amazonaws.com/_attachments/538361/1588286318/azure_ml.JPG)

**Student successfully implements a pipeline in code that will train and predict on the supervised learning algorithm given.**

Nice Job on implementing the pipeline correctly 

* Start and end times are correctly calculated
* accuracy\_score and fbeta\_score is used for calculation from sklearn library
* Dataset is split into size of 300

You can learn more about Pipelines in scikit-learn from the API documentation [Here](http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html)

**Student correctly implements three supervised learning models and produces a performance visualization.**

**Improving Results**

**Justification is provided for which model appears to be the best to use given computational cost, model performance, and the characteristics of the data.**

A well detailed explanation is provided.

RandomForest Classifier is indeed one of the best model for the given task

* Accuracy and F1 score is substantial enough to make the conclusion
* Its really good to see that you made your decision on the basis of test set. That is what will be reflective of real world situation.
* I am happy to see that your conclusion is driven by time analysis as well and not just on the accuracy numbers.

**Student is able to clearly and concisely describe how the optimal model works in layman's terms to someone who is not familiar with machine learning nor has a technical background.**

Really nice.  
You showed that you really understand the model.

**The final model chosen is correctly tuned using grid search with at least one parameter using at least three settings. If the model does not need any parameter tuning it is explicitly stated with reasonable justification.**

Well done on choosing the parameters

parameters = {

'n\_estimators': [200, 500],

'max\_features': ['auto', 'sqrt', 'log2'],

'max\_depth' : [4,5,6,7,8],

'criterion' :['gini', 'entropy']

}

This should lead you to a better optimized model

**Student reports the accuracy and F1 score of the optimized, unoptimized, models correctly in the table provided. Student compares the final model results to previous results obtained.**

All the things have been correctly reported.

Its a good idea to also add a column for Naive Predictor for just seeing how far we have come improving the model

**Feature Importance**

**Student ranks five features which they believe to be the most relevant for predicting an individual's’ income. Discussion is provided for why these features were chosen.**

Very Intuitive. These are some great features.

Just make sure that these are not decided after the results on the next cell as it helps in developing your intuition.

Writing the answer pointwise here gives a better clarity

**Student correctly implements a supervised learning model that makes use of the feature\_importances\_ attribute. Additionally, student discusses the differences or similarities between the features they considered relevant and the reported relevant features.**

An alternative feature selection approach consists in leveraging the power of [Recursive Feature Selection](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html) to automate the selection process and find a good indication of the number of relevant features (it is not suitable for this problem because that is not what is required by the project rubric, though it is generally a very good approach).

Correct implementation using feature\_importances\_ .

It is not about ranking your features. Your intuition for selecting the features should be correct as this intuition can help you in removing some redundant features and thus saving some training time.

**Student analyzes the final model's performance when only the top 5 features are used and compares this performance to the optimized model from Question 5.**

Well done you should see that the model doesn't deteriorate much . This highlights the importance of feature selection