EMORY UNIVERSITY

Introduction to Business Analytics [MSBA]

Homework #2 **(HW2 is Due on Friday, September 21 11.59pm)**

_____Kaitlin Wallis_(Katie)_

(put your full name above (incl. any nicknames); 5 points)

Note: This is an individual homework. Discussing this homework with your classmates is a violation of the Honor Code. If you borrow code from somewhere else, please add a comment in your code to make it clear what the source of the code is (e.g., a URL would sufficient). If you borrow code and you don't provide the source, it is a violation of the Honor Code.

Total grade: _____ out of ___150___ points

1) (15 points) Would you frame the problem of e-mail spam detection as a supervised learning problem or an unsupervised learning problem? Please justify your answer.

I would consider the problem of e-mail spam detection as a supervised learning problem because there is a target that you want to predict: you want to know if an email is spam or not. Because there is a target it is inherently a supervised learning problem. When solving the problem, it would always be a supervised learning method, however, it is possible that while you were exploring your data you wanted to look at whether there were some naturally occurring groups in your data to gain a greater level of understanding of your data. You could do this part of the problem with something like clustering that would be unsupervised, but this part would really be considered part of the data exploration phase. Overall, the problem would be categorized as supervised.

2) (15 points) What is a test set and why would you want to use it?

A test set is a segment of your dataset that you put aside so that it is not being used in the model training or validation. Because this data was not used in any part of the creation of the model, it can give you a good representation of how well the model will predict new unseen data. The test set thus is able to give you generalization performance, performance metrics that are representative of future and new data.

3) (20 points) What are the similarities and differences of decision trees and logistic regression? When might you prefer to use one over another?

Similarities: Both are models that are used for classification and can create generate probability estimates. With both models having many attributes can result in overfitting but both have ways in which automatic feature selection can be used.

Differences: Logistic Regression can look at multiple attributes at the same time while decision trees add attributes one at a time. Logistic Regression creates a single diagonal boundary while Decision Trees allow several perpendicular boundaries that can create several regions.

When to use which: Logistic Regression performs better on small data sets. It also does well when the phenomenon being measured is linear. When features on not linear they can sometimes be transformed but sometimes decision trees do a better job of predicting non-linear patterns (particularly when there is a lot of data). In general, decision trees tend to perform better when there is lots of data. Finally, decision trees are frequently easier for non statistically savvy people to understand.

4) (25 points) You have a fraud detection task (predicting whether a given credit card transaction is "fraud" vs. "non-fraud") and you built a classification model for this purpose. For any credit card transaction, your model estimates the probability that this transaction is "fraud". The following table represents the probabilities that your model estimated for the validation dataset containing 10 records.

Correct at .8?	Correct at .3?	Actual Class (from validation data)	Estimated Probability of Record Belonging to Class "fraud"
T	T	fraud	0.95
T	T	fraud	0.91
F	T	fraud	0.75
T	F	non-fraud	0.67
F	T	fraud	0.61
T	F	non-fraud	0.46
F	T	fraud	0.42
T	T	non-fraud	0.25
T	T	non-fraud	0.09
T	T	non-fraud	0.04

Based on the above information, answer the following questions:

a) **What is the overall accuracy of your model, if the chosen probability cutoff value is 0.3? What is the overall accuracy of your model, if the chosen probability cutoff value is 0.8?**

The overall accuracy of the model with a .3 cutoff point is 80% as the model has correctly identified 8 out of 10 of the instances. The overall accuracy for a model with a .8 cutoff is 70% because it has correctly classified 7 out of 10 instances.

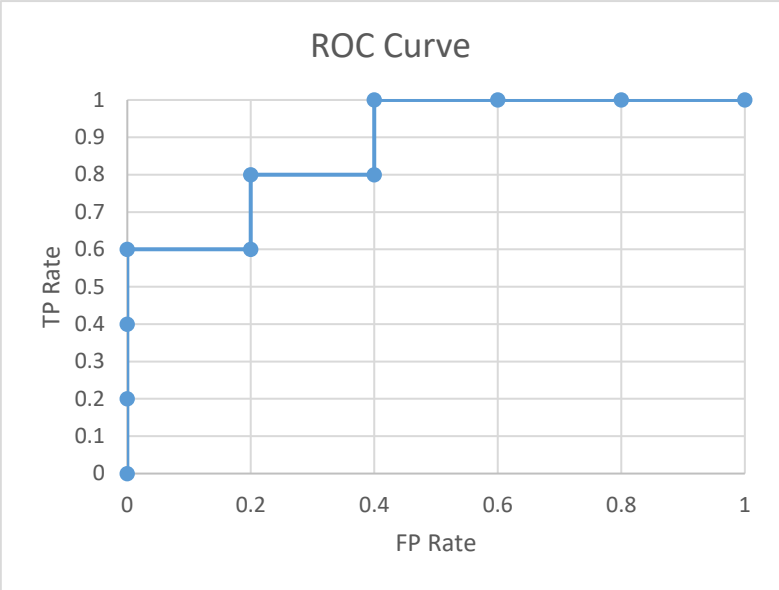
b) **What probability cutoff value should you choose, in order to have Precision fraud = 100% for your model? (Explain.) What is the overall accuracy of your model in this case?**

In order to have a precision of 100% for this validation dataset you would need a cutoff of between 0.67 and 0.75. Choosing a cutoff closer to the 0.75 end of the range would be more conservative and would allow you to be the most sure that you had a precision of 100% while a cutoff on the lower end of the range towards 0.67 would allow you to capture more fraud but would also slightly open up the possibility for a false positive.

c) **What probability cutoff value should you choose, in order to have Recall fraud = 100% for your model? (Explain.) What is the overall accuracy of your model in this case?**

In order to have a recall of 100% for this validation dataset you would need a cutoff of between 0.25 and 0.42. Choosing a cutoff closer to the 0.25 end of the range would be more conservative and would allow you to be the most sure that you had a recall of 100% while a cutoff on the higher end of the range towards 0.42 would allow you to capture more fraud but would also slightly open up the possibility for a false positive.

d) Draw an ROC curve for your model.



5) (70 points) [Mining publicly available data. Please implement the following models both with Rapidminer and Python but explore the data (e.g., descriptive statistics etc.) just with Python]

Please use the dataset on breast cancer research from this link: http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data [Note: Rapidminer can import .data files in the same way it can import .csv files. For Python please read the data directly from the URL without downloading the file on your local disk.**] The description of the data and attributes can be found at this link: http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names . Each record of the data set represents a different case of breast cancer. Each case is described with 30 real-valued attributes: attribute 1 represents case id, attributes 3-32 represent various physiological characteristics, and attribute 2 represents the type (benign or malignant). If the dataset has records with missing values, you can filter out these records using Python and/or Rapidminer before proceeding with the models. Alternatively, if the data set has missing values, you could infer the missing values.**

Perform a predictive modeling analysis on this same dataset using the a) k-NN technique (for k=3) and b) Logistic Regression. Please be specific about what other parameters you specified for your models.

Present a brief overview of your predictive modeling process, explorations, and discuss your results.

Compare the k-NN model with the Logistic Regression model: which performs better? Make sure you present information about the model "goodness" (i.e., confusion matrix, predictive accuracy, precision, recall, f-measure). Please be clear about any assumptions you might make when you choose the best performing model.

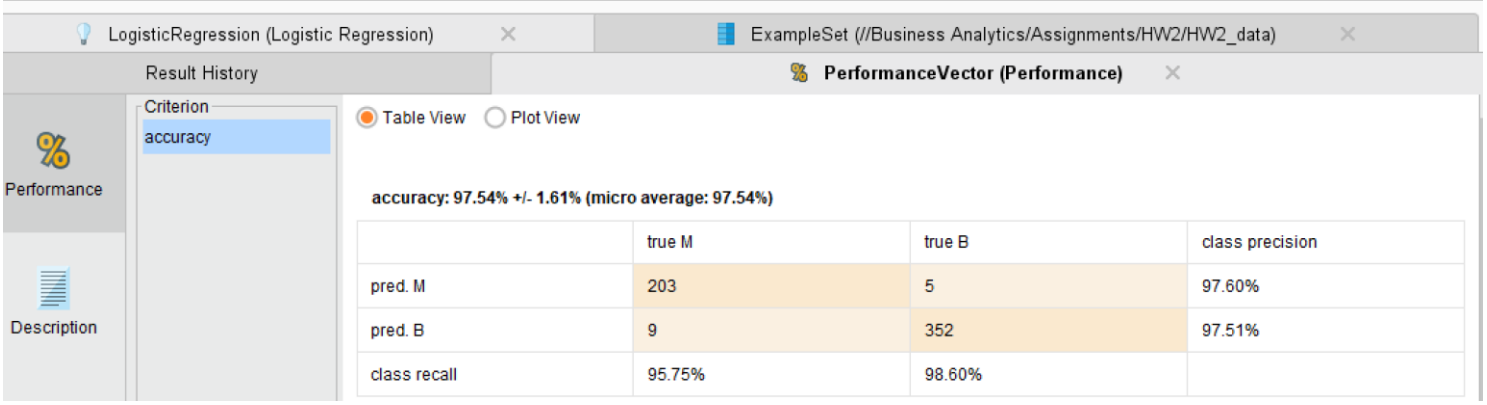
Please show screenshots of the models you have built with Rapidminer and Python, show screenshots of the performance results, and the parameters you have specified.

For this modeling process, I built two models- a K-NN model with a k value of 3 and a logistic regression model that used l1 norm regularization. In both cases, I started building the models using the simple split validation and then also did cross-validation to get an understanding of the generalization performance and the effect of using split validation versus cross validation. When looking at the data, I noticed it was unbalanced with 357/579 being of class B. Given this lack of balance, I choose to stratify the split. Reading the documentation of Cross Validation, I discovered that cross_val_score by default uses stratification if the scorer is a classifier and the target variable is binary or multiclass, stratification is automatically used. With split validation, I had to specify that I wanted to stratify the dataset by setting the stratify parameter equal to my Y value. For the K-NN model, the only parameters I specified was the k = 3 and weighted voting. I left the rest of the parameters as their default. For the Logistic Regression model, I chose to use the L1 penalty. The L1 penalty is useful here because it helps prevent the model from becoming over complex which can lead to overfitting. Once I had the two models built, I was able to look at the generalization performances of cross validation and split validation. One interesting thing I took note of in terms of the generalization performance is that the cross validation accuracy is lower than the split validation accuracy for the Logistic Regression Model and higher than the split validation accuracy for the K-NN model. However, in both cases the split validation accuracy was within 2 standard deviations of that of the cross validation. Most likely, the split validation is randomly using a dataset where the accuracy is unusually high or unusually low. Cross Validation here is giving us a better picture of what is going on because it is giving the distribution of the accuracies.

Here we see the metrics for the Logistic Regression Model:

```
Model Metrics:

Cross Validation Accuracy:  0.951  +/-  1.59
Split Validation Accuracy: 97.2
Positive Precision: 0.96
Negative Precision: 0.98
Positive Recall: 0.96
Negative Recall: 0.98
Positive F-Measure: 0.96
Negative F-Measure: 0.98
```

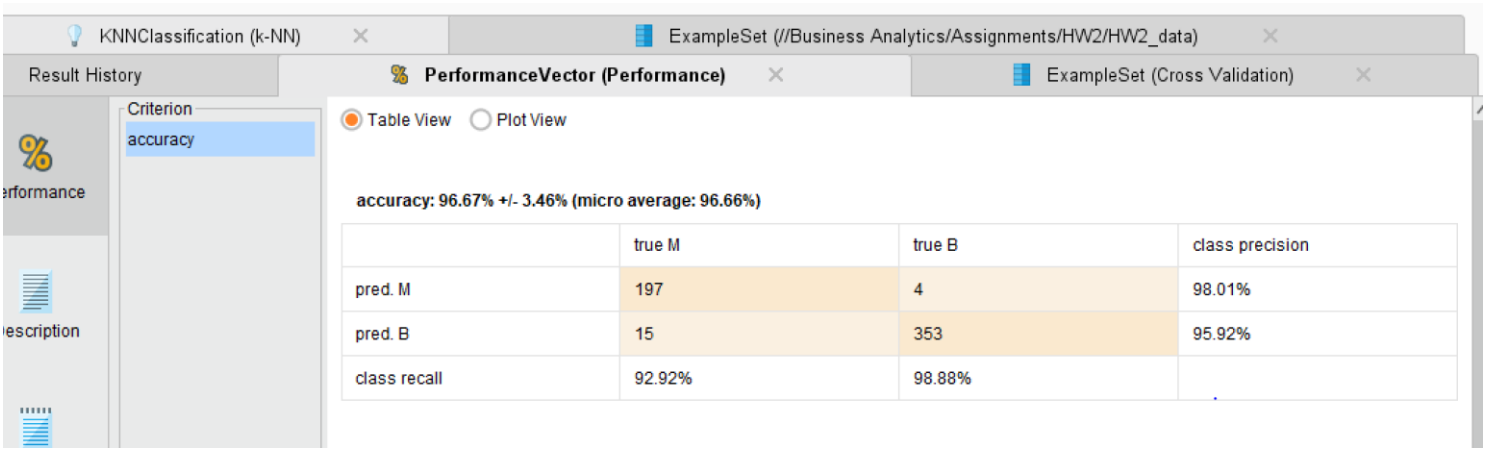


Here are the metrics for the K-NN Model:

```
Model Metrics:

Cross Validation Accuracy:  0.919  +/-  2.39
Split Validation Accuracy: 90.91
Positive Precision: 0.87
```

```
Negative Precision: 0.93
Positive Recall: 0.89
Negative Recall: 0.92
Positive F-Measure: 0.88
Negative F-Measure: 0.93
```

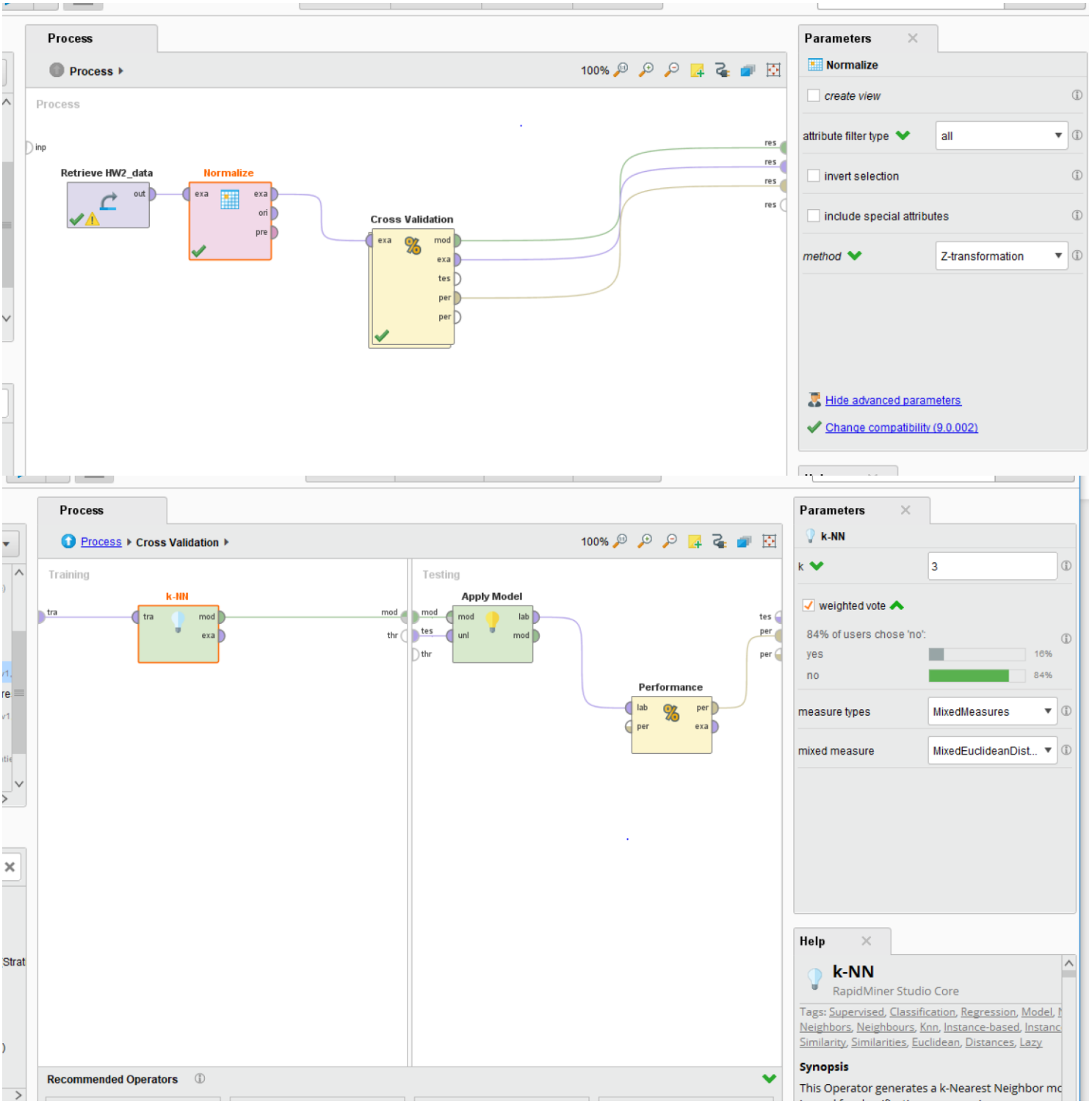


Here we can see that the Logistic Regression model is performing slightly better in terms on accuracy, precision, recall and F-Measure. In both cases we see that the model has a higher negative precision than positive precision, telling us that we are doing slightly better at predicting the negative values than the positive values. Overall, however, our metrics are consistently pretty high. I would say that the Logistic Regression model is the best model to choose, assuming that the cost of false positives are equal. For instance, we see Logistic Regression has a slightly lower class precision for M than the K-NN model and a very slightly lower class recall for B. If differences in costs were substantially different, we may need to take that into consideration when choosing the best model. Overall, however, Logistic Regression appears to perform marginally better.

K-NN:

```
#With Split Validaition
clf1 = KNeighborsClassifier(n_neighbors = 3)
clf1.fit(X_train, y_train)
y_pred= clf1.predict(X_test)

#Create distribution of accuracies using cross validation
acc = cross_val_score(clf1, X, Y, cv = 5)
acc = cross_val_score(clf1, X, Y, cv = 5)
clf1.fit(X_train, y_train)
y_pred= clf1.predict(X_test)
stdev_acc= acc.std().round(4)
mean_acc = acc.mean().round(3)
print("Cross Validation Accuracy: ", mean_acc, " +/- ", stdev_acc*100)
```



Logistic Regression:

```
clf1 = LogisticRegression(penalty = "l1")
acc = cross_val_score(clf1, X, Y, cv = 5)
acc = cross_val_score(clf1, X, Y, cv = 5)
clf1.fit(X_train, y_train)
y_pred= clf1.predict(X_test)
stdev_acc= acc.std().round(4)
```

```
mean_acc = acc.mean().round(3)
print("Cross Validation Accuracy: ", mean_acc, " +/- ", stdev_acc*100)
```