DATA/DSCI 401 Assignment 3

Instructor: Chris Garcia

Instructions:

1. This assignment has two parts and two separate deliverables: a python code file for part 1, and a Jupyter notebook for part 2.

2. The main point of this assignment is to use your knowledge of classification modeling and algorithms to solve a real problem. Googling specific tips and Python how-to's is OK, but please do not search the Internet to look at others' solutions of this problem.

3. In the spirit of 2 above, you may talk to your classmates on how to approach the problems. However, I do not want any code sharing or copying - so please do not look at one another's code until the assignments have been submitted.

4. Please submit **two** links in Canvas: 1) a link to your knn.py file for part 1, and 2) a link to your Jupyter notebook, for part 2.

Part I: Implementing a K-Nearest Neighbor Classifier

For Part I you will implement a KNN classification algorithm according to the following specification. First create a new file called "knn.py" and then inside this file define a Python class** called KNN containing *predict* and *fit* methods. You should be able to create a KNN instance by specifying a value for *k* as well as a distance function (this is similar to creating a Random Forest, where you specify the number of estimators and other parameters). You should also implement a *fit* method (which does nothing in the case of KNN) and a *predict* method. Here is some sample code showing how your class should be used:

```
import knn # Import your knn class

def euclidean_dist(x1, x2):
  # .... compute euclidean distance between vectors x1 and x2

data_x, data_y = ... # Get X and y data
x_train, x_test, y_train, y_test = train_test_split(data_x, data_y, test_size
= 0.3, random_state = 4)

knn = KNN(3, euclidean) # Create a 3-NN algorithm with Euclidean distance
knn.fit(x_train, y_train)
y_hat = knn.predict(x_test)

# .... Print out error metrics between y_test and y_hat
```

Your *predict* method should check to see whether the incoming data *X* is a pandas data frame or a numpy array (i.e. matrix) – if it is a data frame you can convert to a numpy array by using *X.values()* method. This will allow you to then perform your calculations. Also as a reminder, any distance function should take two row vectors (i.e. numpy arrays of equal length) and return the distance between them (examples of distance functions are Euclidean, Manhattan Edge, and Jaccard Distance).

** For more info on creating Python classes, see the following:

1. https://www.learnpython.org/en/Classes_and_Objects
2. https://www.w3schools.com/python/python_classes.asp

You should use your KNN method in your notebook in part 2 as one of the methods you try. Simply import your "knn.py" file (store in the same folder as your notebook and import as shown above) and then use KNN like you would any other algorithm. You may also want to systematically try different values of *k* to arrive at the best model.

Part II: Customer Churn

One of the important applications of classification techniques in marketing analytics lies in predicting whether individual customers will defect to a competitor over a fixed time period (e.g. over the next year). Your goal is to build the best model possible to predict churn. In this problem you will use the "churn_data.csv" file to make these kinds of predictions, and you will implement a Jupyter notebook to accomplish this. Please make sure your Jupyter notebook addresses the questions with appropriate discussion (i.e. you will want to include appropriate discussion around your code).

1) Load the "churn_data.csv" dataset into Python. What is the response variable, and what are the predictor variables?

2) What data transforms are necessary to perform on this data and why?

3) What modeling approaches did you use and why? Describe your model development process, including the different models tried, feature selection methods, and the different transformation techniques you employed. (****NOTE:** Please **do not forget** to use **your KNN algorithm from part 1** as one of your methods).

4) Which error metrics did you use to assess performance and why? What kind of performance did you obtain on the different models you built?

5) Construct the best (i.e. least-error) possible model on this data set. What are the predictors used?

6) Load the dataset "churn_validation.csv" into a new data frame and recode as necessary. Predict the outcomes for each of the customers and compare to the actual. What are the error rates you get based on your selected metrics?

7) Consider the best model you built for this problem. Is it a good model that can reliably be used for prediction? Why or why not?