# IMT 574 Midterm

Problem 1:

*Model 1: GRADIENT DESCENT*

*#isolating independent feature of Age*

*x = derm\_df['Age']*

*#isolating class label of Disease*

*y = derm\_df['Disease']*

*def gradient\_descent(x, y):*

*m1 = 0*

*c1 = 0*

*epochs = 5000 #define number of iterations*

*n = len(x)*

*L = 0.000643 #learning rate*

***Final Output:***

***m 0.006776863593425256, c 2.3224138138359405, cost 2.6737947152441945 iteration***

***4999***

To minimize the cost function and find the global minima, I used 5000 iterations with a learning rate of 0.000643.

For this data set, the optimal intercept is 2.3224138138359405 and the optimal gradient is 0.006776863593425256. This yields the lowest cost function at 2.67

*Model 2: RANDOM FOREST*

*#isolating independent features(clinical and histopathological attributes)*

*X = derm\_df.iloc[0:, 0:33]*

*#isolating class label*

*y = derm\_df.iloc[0:, 34]*

*rfc = RandomForestClassifier(n\_estimators = 100)*

*rfc.fit(X, y.values.ravel())*

***Final Output:***

***A screenshot of a computer

Description automatically generated with medium confidence***

The specified number of trees (n\_estimators) is 100. Random forest accuracy rate is 100%

*Model 3*: kNN

*#isolating independent features(clinical and histopathological attributes)*

*X = derm\_df.iloc[0:, 0:33]*

*#isolating class label*

*y = derm\_df.iloc[0:, 34]*

*#create KNN Classifier model*

*knn = KNeighborsClassifier(n\_neighbors = 9)*

***Final Output:***

*A screenshot of a computer

Description automatically generated with medium confidence*

To decide on the k-value, I used a general rule of thumb of k = sqrt(N)/2. N=358

Based off this, I selected 9 as the k-value

kNN accuracy score is 96.65%

*Model 4: DIVISIVE CLUSTERING, KMeans*

*#isolating independent features*

*X = derm\_df.iloc[0:, 0:33]*

*kmeans = KMeans(n\_clusters = 6)*

*y\_means = kmeans.fit\_predict(X)*

***Final Output:***

*1. [[ 0.00948097 -0.24760133 0.58773905 0.80536067 0.77267663 1.9032821*

*-0.29602932 1.83636914 -0.6014154 -0.55160781 -0.37433593 1.8950372*

*0.06394984 -0.67332089 -0.31323768 0.80959309 0.24502765 -0.30440908*

*-0.09834071 -0.62465179 -0.84923137 -0.61852992 -0.44291707 -0.42693154*

*1.87138022 -0.27045053 1.92887316 0.1445304 1.91733006 -0.20253427*

*-0.23788494 0.57936174 1.95188873]*

*2. [-0.88690947 -0.97008443 -0.77529301 0.45664116 -0.70183421 -0.47084679*

*0.14755002 -0.45429341 -0.60838523 -0.58252617 -0.37433593 -0.46880711*

*-0.13810948 -0.67332089 2.39451784 -0.5045527 0.33753298 0.1686308*

*-0.63612976 -0.62465179 0.74954524 -0.59804577 -0.44291707 -0.48218876*

*-0.46702857 -0.54551102 -0.48081258 -0.67249987 -0.47795742 -0.18671163*

*-0.23788494 -0.18961095 -0.48568554]*

*3. [-0.04249252 -0.0817382 -0.57781658 -0.75409498 -0.70183421 -0.47084679*

*3.52614595 -0.45429341 1.08919166 -0.0337252 1.14853069 -0.46880711*

*-0.34305536 -0.48884471 -0.38129356 0.11824283 -0.43632126 0.38312855*

*-0.04096461 -0.5302951 -0.76286068 -0.61852992 -0.36882159 -0.48218876*

*-0.40856835 -0.54551102 -0.48081258 0.22198083 -0.47795742 3.61649173*

*3.91887126 -0.3967873 -0.45972537]*

*4. [ 0.240811 0.41106437 0.56636798 -0.36898942 -0.03713409 -0.47084679*

*-0.2684523 -0.45429341 1.1196891 1.24681041 0.54421854 -0.46880711*

*-0.34305536 0.0684881 -0.38129356 -1.01830197 0.29071609 0.60631731*

*0.9634511 1.51709515 1.17979437 1.52092623 -0.04303674 1.36527377*

*-0.46702857 1.1138408 -0.46425357 -0.84233798 -0.47795742 -0.23352028*

*-0.23788494 -0.18767653 -0.46187727]*

*5. [ 0.07389339 0.02856306 -0.54244927 -0.23277026 -0.12077817 -0.47084679*

*-0.28023521 -0.45429341 -0.59312875 -0.51267878 -0.29127048 -0.46880711*

*0.33559498 0.13166609 -0.37031182 0.65863772 -0.42982525 -0.36453465*

*-0.4409282 -0.62465179 -0.74715692 -0.59227296 -0.30819803 -0.47030009*

*-0.46702857 -0.34708503 -0.48081258 0.99603088 -0.46847246 -0.21352022*

*-0.21945144 -0.22755635 -0.48848259]*

*6. [ 0.39493788 0.76105181 0.64369931 -0.31297406 0.16569994 -0.47084679*

*-0.18513449 -0.45429341 0.94073437 0.9587872 0.40329809 -0.46880711*

*-0.13810948 1.49852638 -0.38129356 -0.9453003 0.09427821 0.1686308*

*0.54264393 1.2223301 1.02519638 1.14310738 2.04795199 0.60296912*

*-0.46702857 0.50550233 -0.48081258 -0.84233798 -0.47795742 -0.23352028*

*-0.23788494 0.2673957 -0.50491529]]*

*KMeans [4 5 0 5 0 4 1 0 4 4 3 4 4 3 0 4 4 5 0 1 2 4 1 0 1 3 2 1 4 0 3 4 3 4 0 3 4 4 3 4 1 0 4 2 4 0 0 4 3 3 1 3 4 0 4 4 2 3 1 3 4 0 3 4 1 3 4 2 0 1 4 4 4 5 0 1 5 4 4 4 1 5 5 0 3 4 4 4 1 5 0 4 4 1 3 2 4 1 5 4 4 3 4 3 0 3 5 0 1 0 0 1 4 0 4 3 4 1 2 3 3 4 2 0 1 4 5 5 0 1 4 5 4 4 0 5 4 5 5 0 0 0 4 1 4 4 4 5 5 5 1 0 4 0 4 4 4 4 0 2 4 5 5 0 4 0 0 5 5 5 0 5 5 4 0 0 5 4 5 5 2 4 4 4 4 5 0 0 0 5 5 4 0 4 4 4 1 1 1 1 1 5 5 5 5 5 5 5 0 0 0 0 0 0 4 4 4 4 1 1 1 1 1 1 1 4 4 4 4 5 5 3 3 3 3 2 2 3 3 3 3 3 3 3 5 0 0 0 0 0 0 0 4 4 4 4 4 4 2 2 2 4 4 4 3 3 5 3 5 4 4 4 4 4 3 3 4 4 4 0 0 0 0 3 3 3 3 1 1 1 1 1 0 0 0 4 3 3 4 4 4 3 3 3 0 0 0 0 0 3 3 3 3 4 4 5 3 4 0 0 4 5 3 4 4 1 1 3 3 1 1 0 3 1 1 2 2 4 4 2 2 2 3 3 3 1 1 3 3 5 3 4 4 4 4 0 0 3]*

*Model 5: AGGLOMERATIVE CLUSTERING*

*Dendrogram = sch.dendrogram((sch.linkage(X, method = 'ward')))*

*ac = AgglomerativeClustering(n\_clusters = 6)*

*y\_ac = ac.fit\_predict(X)*

***Final Output:***

*Agglomerative clustering [2 0 1 0 1 2 4 1 5 5 0 2 2 0 1 5 2 0 1 4 3 5 4 1 4 0 3 4 5 1 0 2 0 5 1 0 2 5 0 5 4 1 5 3 2 1 1 5 0 0 4 0 2 1 5 2 3 0 4 0 2 1 0 5 4 0 2 3 1 4 5 2 2 0 1 4 0 2 2 2 4 0 0 1 0 5 2 2 4 0 1 5 2 4 0 3 2 4 0 2 2 0 5 0 1 0 0 1 4 1 1 4 2 1 5 0 2 4 3 0 0 2 3 1 4 5 0 0 1 4 4 0 5 2 1 0 5 0 0 5 1 1 2 4 5 2 2 0 0 0 4 1 2 1 2 5 5 5 1 3 2 0 0 1 5 1 1 0 0 0 1 0 0 2 1 1 0 0 0 0 3 2 2 2 2 0 1 1 1 0 0 2 1 2 2 2 4 4 4 4 4 0 0 0 0 0 0 0 1 1 1 1 1 1 5 5 5 5 4 4 4 4 4 4 4 2 2 2 2 0 0 0 0 0 0 3 3 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 5 2 5 5 5 5 3 3 3 5 5 5 0 0 0 0 0 5 2 5 5 5 0 0 5 5 5 1 1 1 1 0 0 0 0 4 4 4 4 4 1 1 1 5 0 0 5 5 5 0 0 0 1 1 1 1 1 0 0 0 0 5 2 0 0 5 1 1 5 0 0 5 5 4 4 0 0 4 4 1 0 4 4 3 3 5 2 3 3 3 0 0 0 4 4 0 0 0 0 2 2 5 5 1 1 0]*

*Chart, histogram

Description automatically generated*

In model 1, I used gradient descent to find the best values of parameters for the *Age* attribute by working to find the lowest cost function. While it is a fairly quick optimization algorithm to use when building a regression model, I find that it can take a bit of trial and error to find the most appropriate learning rate and number of iterations. Learning rate of 0.000643 with 5,000 iterations allowed me to reach the lowest cost function of 2.6738. This was different from my models 2 and 3, in which I used the decision tree and random forest algorithms to classify the disease type. For random forest, the only hyper parameter I had to determine was the n\_estimator value (number of trees). I chose 100 as a standard. KNN required me to determine the appropriate K value, or number of neighbors. I spent some time pondering over my decision but ultimately chose 9, particularly because the data set is larger. I used the equation K = sqrt(N)/2 as a general rule of thumb. So based off the 358 rows, I chose 9 as my K value. My random forest model had an accuracy of 100% whereas my KNN model had an accuracy of 96.65%. This was expected considering the random forest’s ensemble methods in which it operates by constructing many decision trees to minimize overfitting, which is likely why the accuracy rate was higher (additionally, train/test was not utilized per instruction).

The last two models I used were clustering algorithms. My model 4 utilized divisive clustering using KMeans. The hyper parameter was the number of clusters in which I specified 6 because there are 6 types of diseases in the dataset. Similarly, I specified 6 clusters in my model 5, which used agglomerative clustering. In looking at the clustering assignments from using kNN (top down) approach and the agglomerative (bottom up) approach, it is clear that the outputs are quite different. This is attributed to the distinctive ways in which each type of clustering algorithm makes different decisions on *how* to split the data. There are situations in which one clustering method may be better than the other. Agglomerative clustering is good for exploration where the goal is to better understand how things are related. Divisive clustering is a good application when we already know what clusters we are looking for. Considering that we already know that there are 6 types of diseases (or 6 clusters), I would say that kNN would be most appropriate out of the two in this scenario.

By applying different models on the same data set, we can see that outputs can vary based on the model that is used. While it is very important to understand how to use different algorithms, it is equally important to understand *when* to use then. This requires the data scientist/analyst to clearly understand the data they are working with and to identify what the main goal is (i.e. exploration, prediction, classification, etc.).

Problem 2:

1. *How does income inequality relate to the number of hate crimes and hate incidents?*

To address this question, I used linear regression because the hate crime label is a continuous value. I used Gini index score to measure income inequality and reviewed the OLS Regression Results for both labels ‘hate\_crimes\_per\_100k\_splc’ and ‘avg\_hatecrimes\_per\_100k\_fbi’

When comparing attribute gini\_index to labels hate\_crimes\_per\_100k\_splc and avg\_hatecrimes\_per\_100k\_fbi labels, the p-value is > 0.05 for both, which means that there is a statistically significant relationship between Gini index score and crime. Based off the positive coefficient values, we can conclude that the Gini index score and crime rate/incidents have a positive relationship.

***Final Output:***

*Dep. Variable: hate\_crimes\_per\_100k\_splc*

*==============================================================================*

*coef std err t P>|t| [0.025 0.975]*

*------------------------------------------------------------------------------*

*const -1.2754 0.533 -2.394 0.021 -2.350 -0.201*

*gini\_index 3.1512 1.168 2.698 0.010 0.796 5.507*

*==============================================================================*

*Omnibus: 24.569 Durbin-Watson: 1.938*

*Prob(Omnibus): 0.000 Jarque-Bera (JB): 40.306*

*Skew: 1.661 Prob(JB): 1.77e-09*

*Kurtosis: 6.235 Cond. No. 58.4*

*==============================================================================*

Linear equation using gini\_index to predict hate\_crimes\_per\_100k\_splc:

**hate\_crimes\_per\_100k\_splc = -1.2754 + 3.1512\*x**

*Dep. Variable: avg\_hatecrimes\_per\_100k\_fbi*

*==============================================================================*

*coef std err t P>|t| [0.025 0.975]*

*------------------------------------------------------------------------------*

*const -1.5307 0.475 -3.222 0.002 -2.489 -0.573*

*gini\_index 3.7675 1.042 3.617 0.001 1.667 5.868*

*==============================================================================*

*Omnibus: 12.836 Durbin-Watson: 2.310*

*Prob(Omnibus): 0.002 Jarque-Bera (JB): 16.251*

*Skew: 0.917 Prob(JB): 0.000296*

*Kurtosis: 5.302 Cond. No. 58.4*

*==============================================================================*

Linear equation using gini\_index to predict avg\_hatecrimes\_per\_100k\_fbi:

**avg\_hatecrimes\_per\_100k\_fbi = -1.5307 + 3.7675\*x**

1. *How can we predict the number of hate crimes and hate incidents from race/nature of the population?*

To address this question, I used linear regression because the hate crime label is a continuous value. I began by plugging in all attributes (omitting Gini index score) as X then explored the relationships between hate\_crimes\_per\_100k\_splc and avg\_hatecrimes\_per\_100k\_fbi (assigned as y). I then compared p-values to identify which attributes were most statistically significant.

*For label hate\_crimes\_per\_100k\_splc, the top 3 attributes with the lowest p-values were: share\_voters\_voted\_trump, share\_unemployed\_seasonal, median\_household\_income*

***hate\_crimes\_per\_100k\_splc = -0.0038 - (1.3202\*x1) + (3.5613\*x2) - (4.89e-06\*x3)***

*For label avg\_hatecrimes\_per\_100k\_fbi, the top 3 attributes with the lowest p-values were: share\_voters\_voted\_trump, share\_unemployed\_seasonal, share\_non\_citizen*

***avg\_hatecrimes\_per\_100k\_fbi = 0.4265 - (-1.3202\*x1) + (3.5613\*x2) + (0.3391\*x3)***

Reflection:

Dealing with data related to people or groups of people can have various ethical implications, and it is our responsibility to consider these potential issues through our analysis process. Like with many things, findings from data often must be taken with a grain of salt in order to avoid jumping to inaccurate conclusions. For example, a person reviewing the outputs of this linear regression model may assume that people who are Trump supporters, employed, and are citizens commit less crimes simply because lower crime rate/incidents were associated with these attributes. Sometimes, ML methods are not enough on their own to address complex issues such as this one. We may run the risk of over-generalizing.

1. *How does the number of hate crimes vary across states? Is there any similarity in number of hate incidents (per 100,000 people) between some states than in others — both according to the SPLC after the election and the FBI before it?*

I used a clustering method through KMeans to address this question. I assigned hate\_crimes\_per\_100k\_splc and avg\_hatecrimes\_per\_100k\_fbi as X. I specified 3 clusters.

*#isolate and scale attributes*

*X = hc\_df[['hate\_crimes\_per\_100k\_splc', 'avg\_hatecrimes\_per\_100k\_fbi']]*

*#scale data*

*scaler = StandardScaler()*

*scaler.fit\_transform(X)*

*#K-Means*

*kmeans = KMeans(n\_clusters = 3)*

*y\_means = kmeans.fit\_predict(X)*

***Final Output:***

*[[0.09451794 0.12887445]*

*[0.30127306 0.299291 ]*

*[1. 1. ]]*

*KMeans [0 0 0 0 0 1 1 0 2 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0]*

*Chart, scatter chart

Description automatically generated*

Based on the clustering assignments, Washington DC is shown to have the highest crime rate and incident (cluster 2). Colorado, Connecticut, Kansas, Maine, Maryland, Michigan, Nevada, and West Virginia follow after (cluster 1).

References:

*5.2 Agglomerative Clustering* [Canvas]. University of Washington. <https://canvas.uw.edu/courses/1516682/pages/5-dot-2-agglomerative-clustering?module_item_id=14325975>

*5.3 Divisive Clustering* [Canvas]. University of Washington.

<https://canvas.uw.edu/courses/1516682/pages/5-dot-3-divisive-clustering?module_item_id=14325976>