**Assignment 2: Learning and Comparing classification models for a dataset**

**Dataset used:** Page Block Classification

**Dataset description/objective:** The main objective of the data set is to classify all blocks of a page layout of the document that has been detected by the segmentation process.

**Dataset attribute:**

1. Height of the block.
2. Length of the block.
3. Area of the block (height \* length);
4. The eccentricity of the block (length/height);
5. Percentage of black pixels within the block (blackpix / area);
6. Percentage of black pixels after the application of the Run Length Smoothing Algorithm (RLSA) (blackand / area);
7. Mean number of white-black transitions (blackpix / wb\_trans);
8. The total number of black pixels in the original bitmap of the block.
9. The total number of black pixels in the bitmap of the block after the RLSA.
10. The number of white-black transitions in the original bitmap of the block.

**Dataset classification:**

* The data instances in the data set will fall under one of these classifications:
  + Text
  + Horizontal line
  + Vertical line
  + Picture
  + Graphic

**Other dataset info:**

* There’s no missing value in the dataset
* 89% of the dataset is classified as text and the rest falls under the other 4 classifications

**Programming Language used:** Python

**Library Utilized:**

* Pandas: for dataset manipulations and storing
* Numpy: mathematical application on the dataset
* Sci-kit learn: provides the default classification models and other data analysis technique
  + **from** sklearn.model\_selection **import** KFold: for 10 fold cross-validation
  + **from** sklearn.preprocessing **import** StandardScaler: for data normalization or standardization
  + **from** sklearn.model\_selection **import** train\_test\_split: for splitting between test and training data
  + **from** sklearn.neighbors **import** KNeighborsClassifier: for KNN classifier application
  + **from** sklearn.metrics **import** classification\_report, confusion\_matrix: for confusion matrix and classification report of the model
* Mat plot lib: for data visualization
* Keras: for neural network classifier application

**Neural Networks - ANN**

I utilized the Keras python library to create an ANN. Using the 10-k fold cross-validation for three different algorithms relu, elu, and tanh. These three models have an average accuracy of 95.76%, 95.87%, 95.89%. Compare to the other classifiers the ANN has the highest accuracy. The output layer has a softmax activation function. And the loss function was categorical\_crossentropy because there were 5 different classes in the dependent variable. The three different algorithms have almost the same accuracy, but the tanh with softmax gave better accuracy.

I chose these three different activation functions because it was giving high accuracy. For the training, I used 100 epochs and batch\_size of 10. This did the training for 10 rounds and 100 epochs.

One important observation I noticed in the dataset is the dependent variable has more class 1 than any other class. This can be one of the reasons why the accuracy is high for the ANN.

**K Nearest Neighbor - kNN (Chantha Mak)**

How I approach this model classifier:

For kNN, I used the default Euclidean distance kNN classifier that is provided by sklearn library to do this assignment. The main deciding factor of this algorithm is the k value which is a parameter that refers to the number of nearest neighbors to include in the majority of the voting process. K-value has a big impact on the accuracy of the model so choosing the right k was challenging. In order to pick the right k value, I compute a series of error detecting models using the elbow method. Elbow methods take the average rate of prediction of each model that is not equal to the test value for all possible k value in a given range. In my case, I choose the k value range of 1 to 40. I ran a loop iteration of 1 to 40 using the elbow method and append the error rate for k-value based model. After the iteration, I plotted the error rate of each k value using matplotlib.

From the plot graph, k value of 3 gives out the lowest error rate which means using this k value would give me the best accuracy. Even though a small k value is prone to noises in the dataset, I decided to choose a small k value still because as the k value increases the accuracy gets worse. Since there’s about 5 thousand data instances, choosing a bigger k value will increase the bias and more time consuming so I opted for a smaller k-value.

Once I picked the k value of 3, I ran 10 fold cross-validation for better accuracy of the data. Inside the 10 fold cross-validation iteration, I pre-processed the data using normalization of a range [0,1]. Then I would train the data, compute the accuracy and append the accuracy to a list. At the end of the iteration, I computed the mean of the accuracy list using numpy and that would be the average accuracy of the knn model. I get an accuracy of 94.7%.

**Decision Trees**

For this project, I decided to utilize Python as my programming language, because I felt that it provided much more support compared to R.

My approach for this model classifier heavily relies on sci-kit learn.

The libraries from sci-kit that i used include the following:

* from sklearn.preprocessing import StandardScaler, LabelEncoder
* from sklearn.model\_selection import train\_test\_split
* from sklearn.metrics import accuracy\_score
* from sklearn.model\_selection import KFold

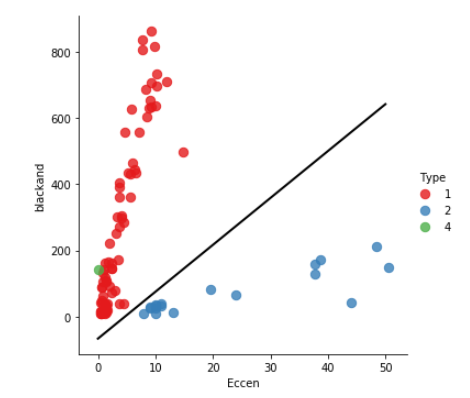
For the decision tree, I used the.DecisionTreeClassifier function provided by sklearn. The parameters included the criterion and the min\_samples\_split. For the criterion, I tested both metrics, entropy, and Gini. After seeing the results, entropy seemed to be the parameter that yielded better results in terms of accuracy. The default min\_sample split is set to 2. After trying multiple amounts of minimum sample splits, I finally settled on 30 to be the best amount, which gave the best results. The best accuracy I got was with entropy and 30 min split, which ranged from 86 to 88. The mean accuracy was 87.64 %.

**Support Vector Machine**

The tools I used for the SVM model were python along with some machine learning libraries. I used pandas to load in my data, numpy to assist in some of the training methods, sklearn for the support vector machine model and matplotlib to create a sample plot.

Our data contained 10 unique features making it highly dimensional and hard to visualize on a plot but SVM excels at multi-dimensional data because of the kernel trick which is a way of computing the dot product of two vectors x and y in some feature space. SVM has 4 different kernel types to choose from which are linear, polynomial, radial basis function and sigmoid. I found the highest accuracy with linear and radial basis function resulting in an accuracy rate of 93% for linear and 92% for radial. When testing Polynomial and sigmoid my accuracy rate went down to around 80%. Even though linear had a higher accuracy rating I found that RBF would be a much better option as our data is not linear and is highly dimensional. After attempting multiple different methods I ended up sticking to RBF and began to tune the model by changing the gamma in powers of 2. I found that if the gamma is set to high our accuracy begins to get negatively affected and so after multiple iterations I found the gamma was set to 8. The training time for my model was taking around 4 to 7 minutes every run which is one of the downsides of SVM since it is very computationally expensive. Once I normalized my values between 0 and 1 I actually reduced that time to less than 10 seconds of training time. I felt my model was at a good place and then proceeded to run 10 fold cross validation and resulted with a 93% accuracy rating for my model.

At this point I wanted to visualize my model in a 2D plot by picking 2 features that seemed to best classify my data. I picked blackand which is the percentage of black pixels within the block and eccentricity which is the block length / width. For this experiment I then switched to a linear kernel type as now this would best classify my data because we are in 2 dimensions. I then plotted the first 100 data points and drew a hyperplane which classified the data almost perfectly. I found that these 2 variables classify text and horizontal lines very well but is not enough to classify pictures. The green plot shown below in the figure represents a picture and is classified along with text. By adding all our other data features we can better classify between a picture and text and so it seems that these two features share similar values.



**Conclusion and Classifications comparison**

Above is the accuracy report of the classification models used for this assignment. The neural network was able to edge out other classification models with an accuracy of 95.89 while the Decision tree has the lowest accuracy of 87.64. None of the classification models really outperform one another since all of them were able to achieve a relatively high accuracy rate. This is because of the distribution of the dataset. Since 89% of the dataset is classified as text, the dataset is highly biased.