HOMEWORK 2

Encinas, Michael A

September 16 of 2024

Intro

For my assignment I am given two datasets, test.csv and train.csv. The data consist in total of 59 attributes, 58 being attributes and 1 being the output target variable. The target variable is either spam or not spam for e-mails. More in detail will be given on these attributes.

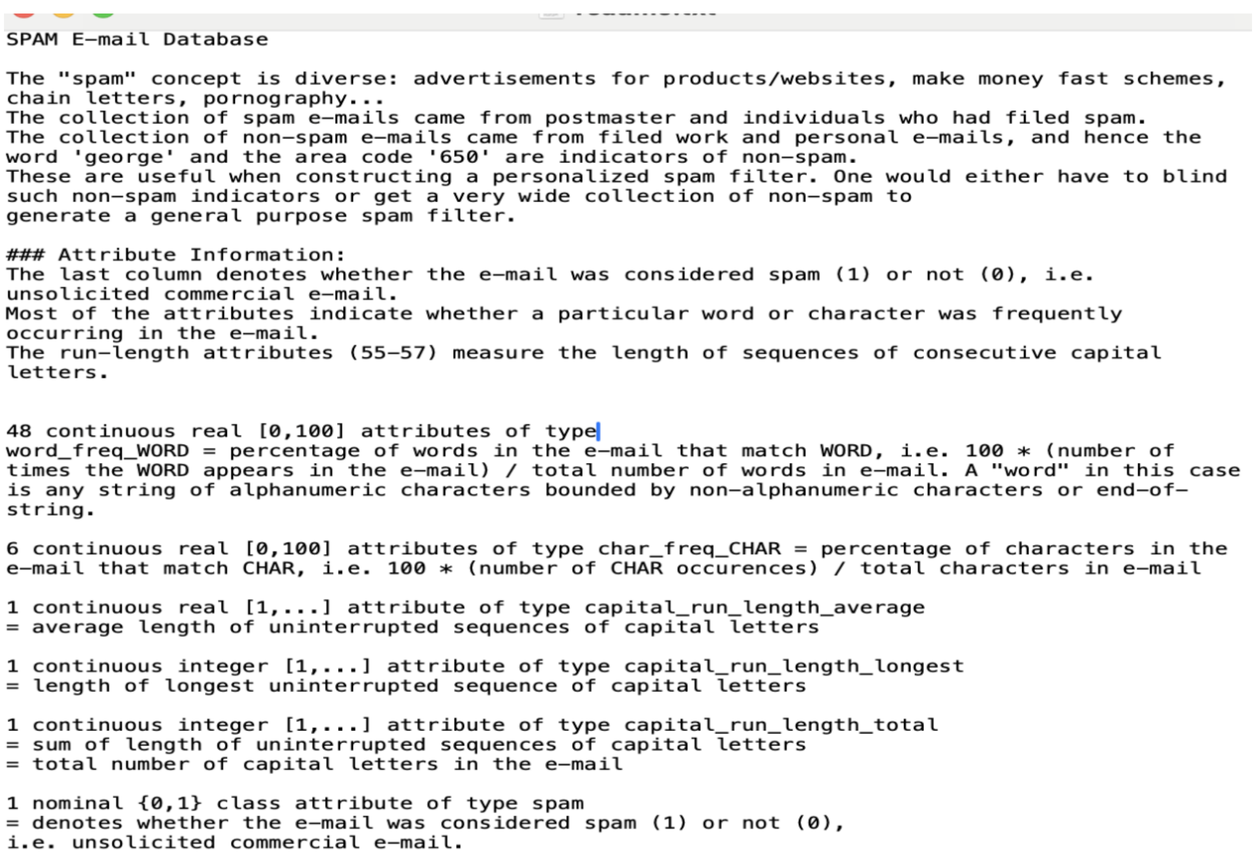
I must gather my data and build three models to predict the correct output targets on my test.csv file. I will be applying the following classification methods to the SPAM e-mail data. KNN Binary Classifier, Logistic Regression Classifier, and Linear Support Vector Machines Classifier.

I will use k-fold cross validation (K = 5) to identify the parameters that optimize generalization for each method. K-fold will be used on the training data. I will be vary the parameters of K in KNN, C in Logistic Regression, and C in Linear Support Vector Machines to see if my CV (Cross Validation accuracy) accuracy increases or decreases. I will average out the CV accuary with each parameter on each model and the best CV accuracy parameter for each model will be selected to go on and test on the testing data. I will also be looking at explainability, and time required on train data to see what the best model is as well.

Once I select my parameters for my models, I will run the train data on that parameter and run it against the test file. I will be given generalization scores, time for train and testing, explainability as well as that parameters CV score to see if I get any insights.

I will check to see where overfitting or underfitting is occurring if it occurs as well as compare my three models to each other to find out what is the best model for my dataset. I will be looking into what my three models are correctly or incorrectly identify target variables to see if I can see any patterns.

Provided below is the text file given to me by professor that explains my SPAM E-Mail Database



```
SPAM E-mail Database

The "spam" concept is diverse: advertisements for products/websites, make money fast schemes,
chain letters, pornography...
The collection of spam e-mails came from postmaster and individuals who had filed spam.
The collection of non-spam e-mails came from filed work and personal e-mails, and hence the
word 'george' and the area code '650' are indicators of non-spam.
These are useful when constructing a personalized spam filter. One would either have to blind
such non-spam indicators or get a very wide collection of non-spam to
generate a general purpose spam filter.

### Attribute Information:
The last column denotes whether the e-mail was considered spam (1) or not (0), i.e.
unsolicited commercial e-mail.
Most of the attributes indicate whether a particular word or character was frequently
occurring in the e-mail.
The run-length attributes (55-57) measure the length of sequences of consecutive capital
letters.


48 continuous real [0,100] attributes of type
word_freq_WORD = percentage of words in the e-mail that match WORD, i.e. 100 * (number of
times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case
is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-
string.

6 continuous real [0,100] attributes of type char_freq_CHAR = percentage of characters in the
e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average
= average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest
= length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total
= sum of length of uninterrupted sequences of capital letters
= total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam
= denotes whether the e-mail was considered spam (1) or not (0),
i.e. unsolicited commercial e-mail.
```

Quick Summary of Models (KNN, LR, SVM).

KNN is an instance-based learning algorithm where the data makes classes based on neighbors and when a new data point is added. It checks the K-nearest neighbor data points and then puts that data point in that class with the closest neighbors.

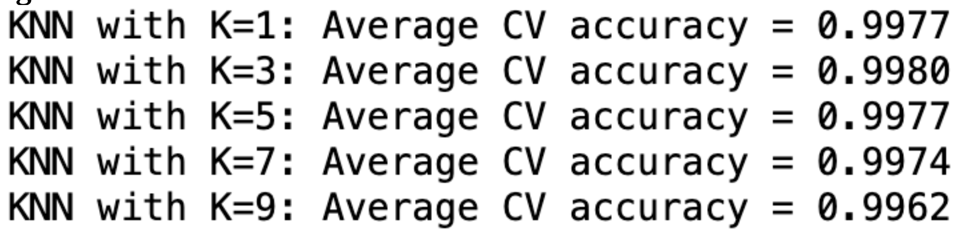
Logistic Regression is a linear model used for binary classification where the relationship between input features and the output probability of a class is modeled from using a sigmoid function. A regularization term, C, is used to control the model complexity.

Support Vector Machine is like LR; however, it uses C to control the tradeoff between achieving a large margin to minimizing classification errors. SVM is fining the optimized hyperplane that separates each class.

KNN

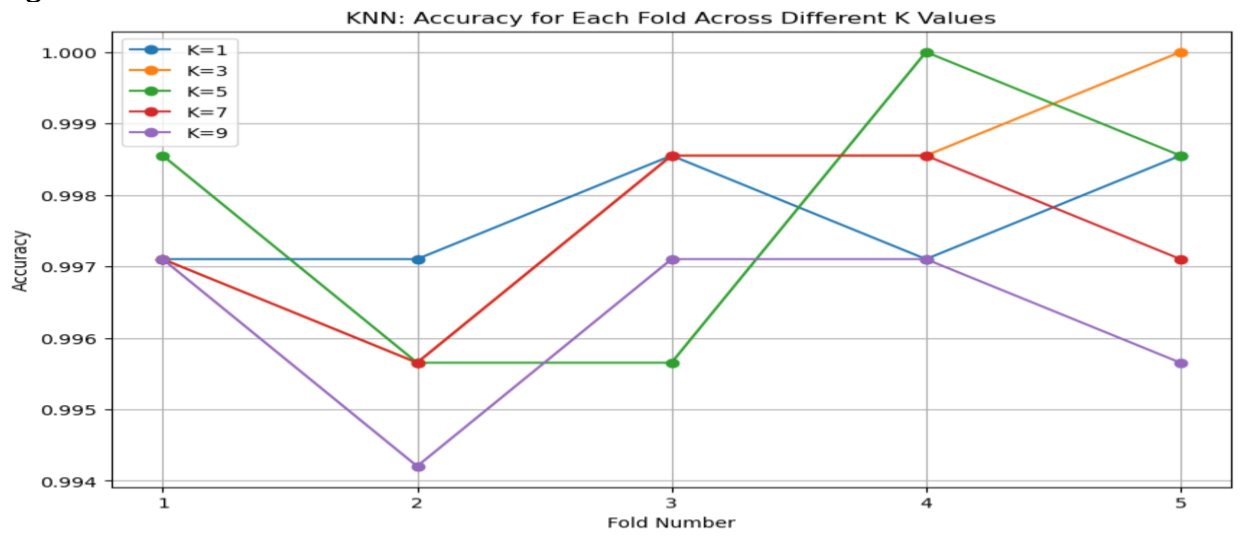
I used K-fold = 5 for KNN and selected 5 unique neighbors to test my model for the different parameters. When using the training and validation for my KNN I got the output below in Figure 1. Figure 1 gets an average of those 5 folds in each neighbor parameters and averages it out. Figure 2 shows the fold number of each parameter and what that specific cv accuracy was.

Figure 1



```
KNN with K=1: Average CV accuracy = 0.9977
KNN with K=3: Average CV accuracy = 0.9980
KNN with K=5: Average CV accuracy = 0.9977
KNN with K=7: Average CV accuracy = 0.9974
KNN with K=9: Average CV accuracy = 0.9962
```

Figure 2



Based on the output in Figure 1 and 2, KNN = 3 has the best CV accuracy with .9980. I wanted to see what the times were from running KNN with these parameters and I had an output that showed K=3 had the fastest run time of .08 seconds. Figure 3 and Figure 4 show this.

Figure 3

```
KNN with K=1: Average CV accuracy = 0.9977, Time taken = 0.11 seconds
KNN with K=3: Average CV accuracy = 0.9980, Time taken = 0.08 seconds
KNN with K=5: Average CV accuracy = 0.9977, Time taken = 0.09 seconds
KNN with K=7: Average CV accuracy = 0.9974, Time taken = 0.09 seconds
KNN with K=9: Average CV accuracy = 0.9962, Time taken = 0.09 seconds
```

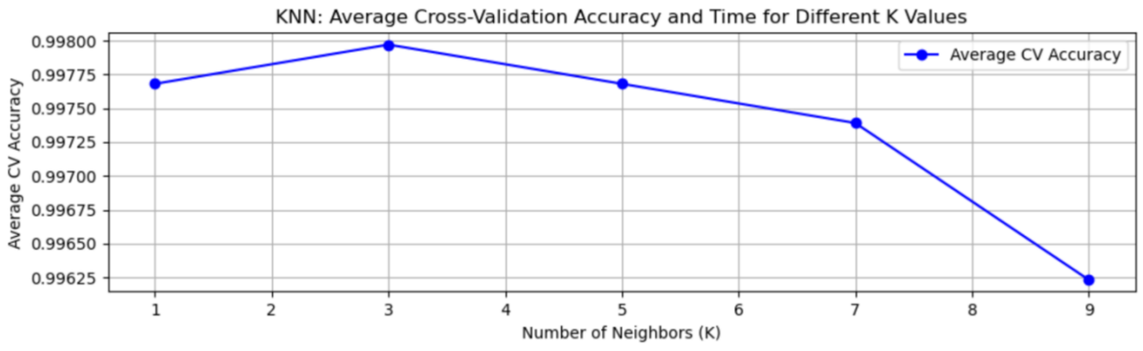
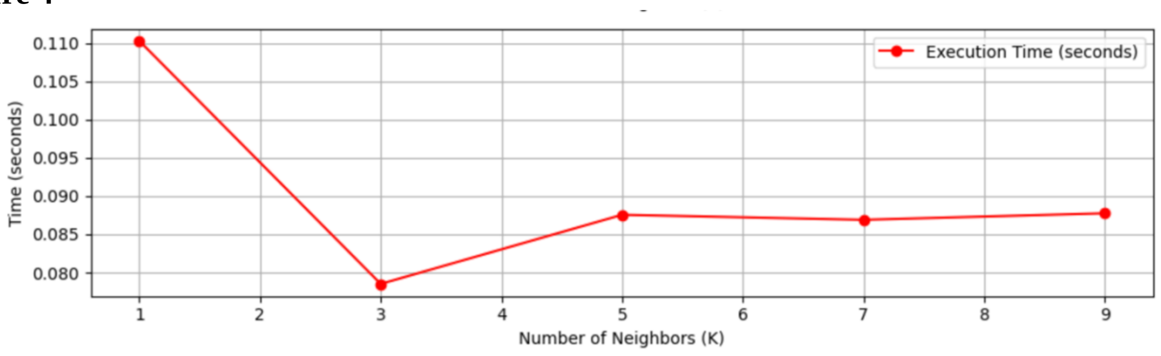


Figure 4



Based on this I will be selecting K = 3 for my parameter to run my model later on against the test data.

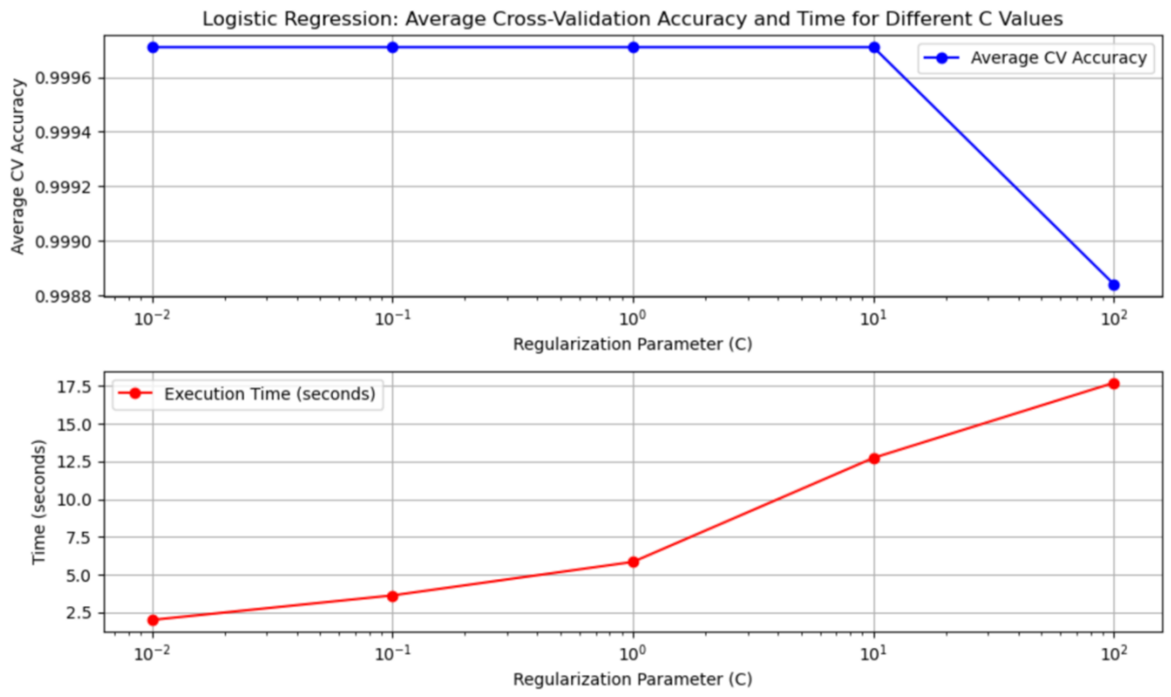
Logistic Regression

I continued to do K-fold = 5 on LR and I varied my c values. C_values are [.01, 0.1, 1, 10, 100). I continued to print the average cv accuracy with its time taken to completion. Figure 5 is my output. Figure 6 is a visualization of my output.

Figure 5

```
Logistic Regression with C=0.01: Average CV accuracy = 0.9997, Time taken = 2.00 seconds
Logistic Regression with C=0.1: Average CV accuracy = 0.9997, Time taken = 3.63 seconds
Logistic Regression with C=1: Average CV accuracy = 0.9997, Time taken = 5.85 seconds
Logistic Regression with C=10: Average CV accuracy = 0.9997, Time taken = 12.74 seconds

 Logistic Regression with C=100: Average CV accuracy = 0.9988, Time taken = 17.72 seconds
```

Figure 6



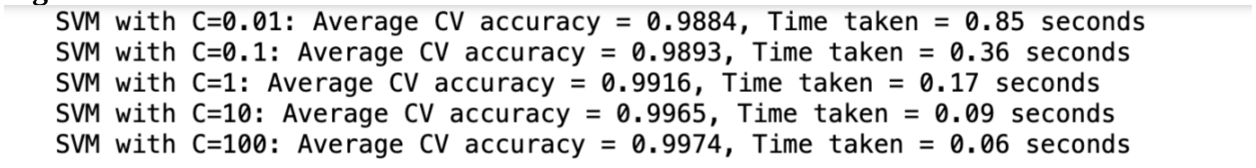
For LR, the times tend to be lower at c = .01, .1, and 1. For higher C the times tend to be high at 12.74 seconds and 17.72 seconds. Based on execution time and cv accuracy, the best parameter on this model is c = .01 since it has the lowest time at 2.0 seconds and an accuracy of .9997.

All the cv accuracy times on every parameter are high and the same besides c = 100 which drops off at .9988 accuracy. Reference Figure 6.

SVM

I continued for K-fold and used the c-values of .01, .1, 1, 10, and 100 for SVM. In figure 7 you get my output with train speed and cv accuracy on every parameter. Figure 8 and Figure 9 will visualize my parameters.

Figure 7



```
SVM with C=0.01: Average CV accuracy = 0.9884, Time taken = 0.85 seconds
SVM with C=0.1: Average CV accuracy = 0.9893, Time taken = 0.36 seconds
SVM with C=1: Average CV accuracy = 0.9916, Time taken = 0.17 seconds
SVM with C=10: Average CV accuracy = 0.9965, Time taken = 0.09 seconds
SVM with C=100: Average CV accuracy = 0.9974, Time taken = 0.06 seconds
```

Figure 8

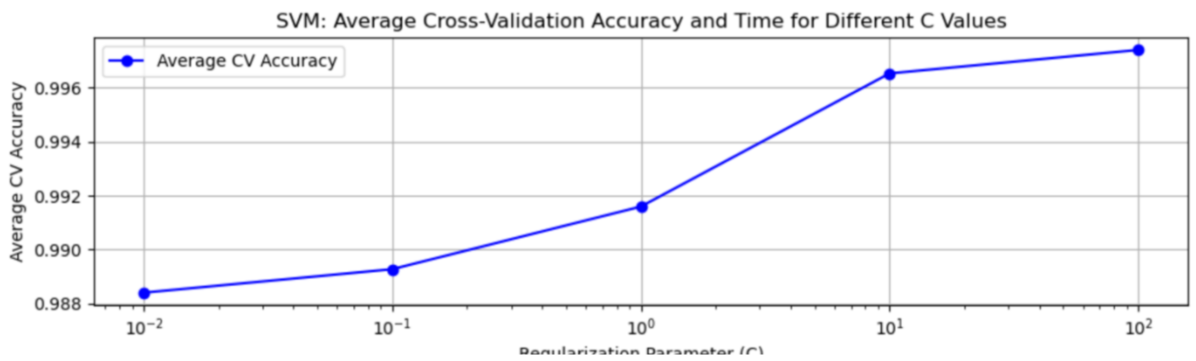
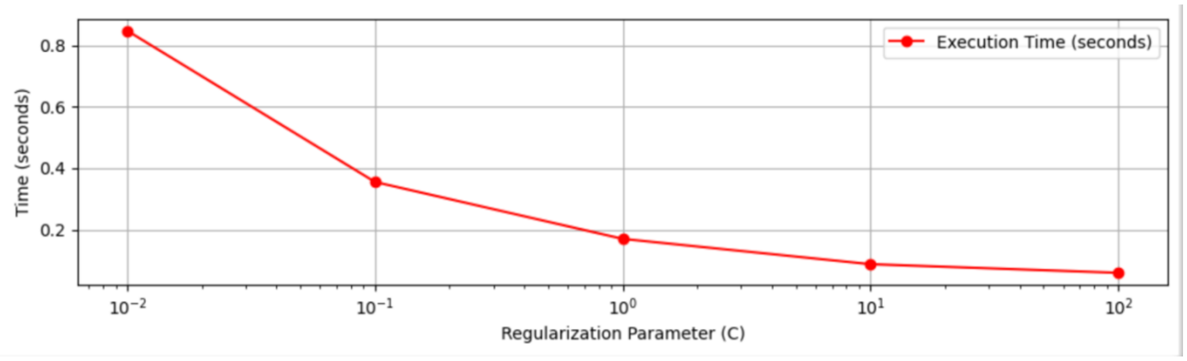


Figure 9



Based on my results in Figure 7,8, and 9, The higher c's (1,10,100) were faster in times compared to the lower c's (.01, .1). The accuracy rate among them were similar but the higher c's (1,10,100) had slightly better accuracy.

Based on the time and accuracy I chose C= 100. (9974 accuracy, time taken 0.06 seconds).

Here is my top parameters for each model

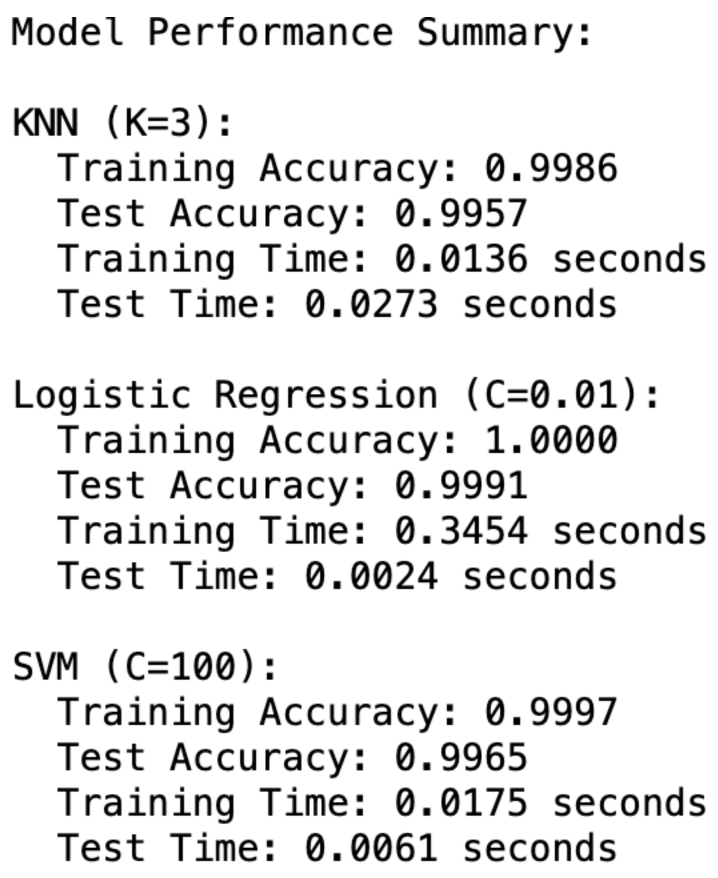
KNN with K=3: Average CV accuracy = 0.9980, Time taken = 0.08 seconds
Logistic Regression with C=0.01: Average CV accuracy = 0.9997, Time taken = 1.56 seconds SVM with C=100: Average CV accuracy = 0.9974, Time taken = 0.06 seconds

Final Model Evaluation Train vs Testing

Now I will be using the best parameters for my models and training the entire train dataset with them. Then I will test them on test datasets and compare my accuracy, train/test speed, and explainability.

I will compare each model to each other as well as their cv accuracy scores compared to test scores and make a final determination which one is the best model for my data. Underfitting and overfitting will also be analyzed if occurring.

Figure 10



```
Model Performance Summary:

KNN (K=3):
  Training Accuracy: 0.9986
  Test Accuracy: 0.9957
  Training Time: 0.0136 seconds
  Test Time: 0.0273 seconds

Logistic Regression (C=0.01):
  Training Accuracy: 1.0000
  Test Accuracy: 0.9991
  Training Time: 0.3454 seconds
  Test Time: 0.0024 seconds

SVM (C=100):
  Training Accuracy: 0.9997
  Test Accuracy: 0.9965
  Training Time: 0.0175 seconds
  Test Time: 0.0061 seconds
```

In figure 10, it shows my results when I run the models with their parameters. Figure 11 and 12 show comparisons between all three models in which includes cv scores on those parameters. Figure 10,11, and 12 shows cv score, train score, test score, train and test times.

Figure 11

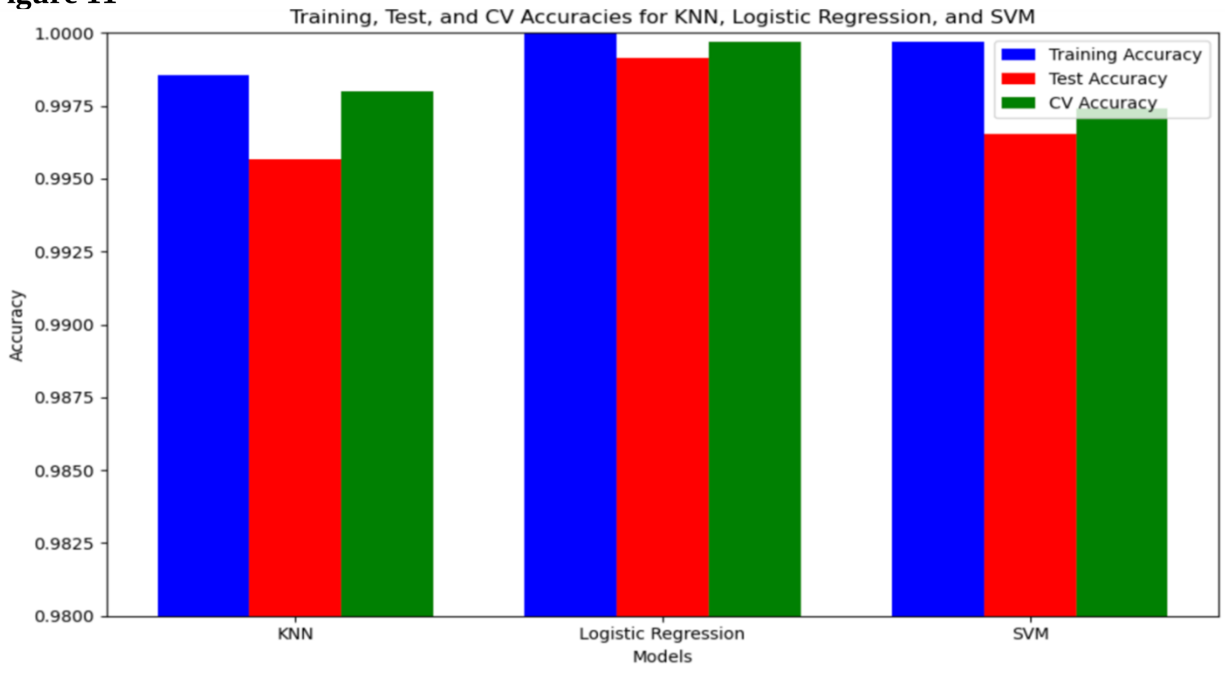
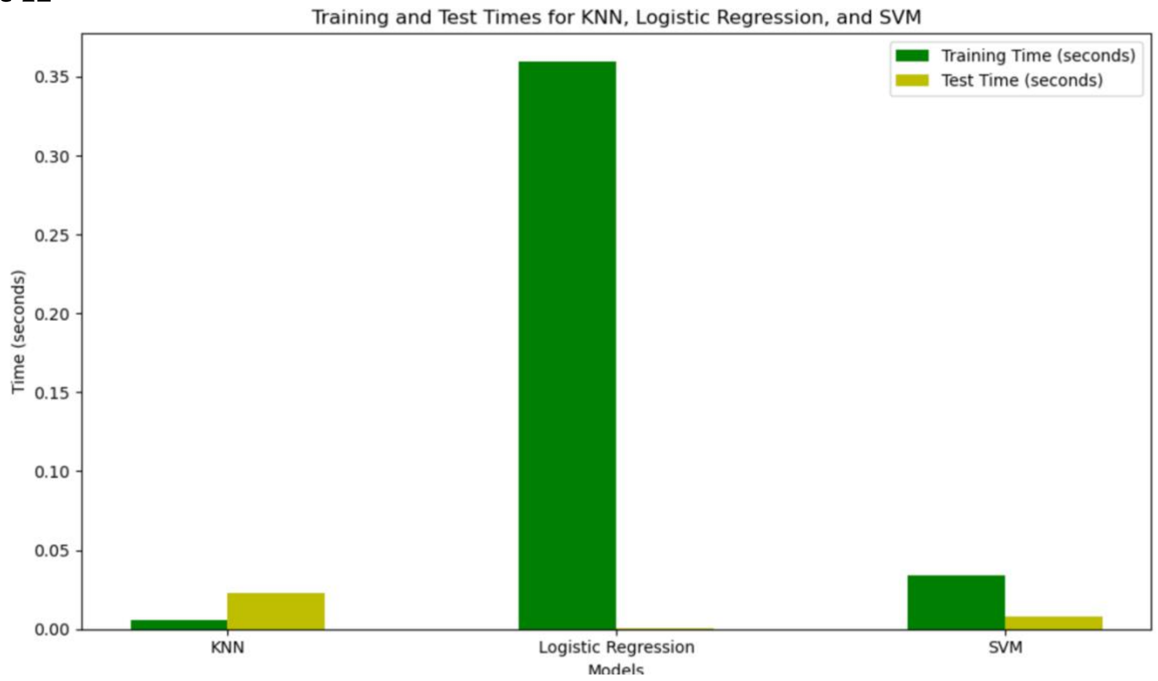


Figure 12



In Figure 12, LR had the fast test time compared to KNN and SVM but also had the slowest training time compared to KNN and SVM. KNN had the fasters train time but the highest train time. SVM a little above slower training time on its model compared to KNN and a little above LR test time.

I'm using figure 10 and 11 as reference, when comparing the accuracy score of test and train, LR had the best r2 score with 1.0 as well as the best test accuracy score with .9991. However, I believe this might be overfitting since the train score is 1.0 which is perfect. KNN has a test score of .9957 and a train score of .9986. I believe this is good generalization since numbers are close to each other. The same goes to SVM, however, SVM has a higher score on training and testing (.9997, .9965). All three models have good accuracy scores, and the range is so small between them on change.

When looking at their CV scores to see if they are like testing score (Figure 11). All three models have very similar CV scores compared to their testing scores. There is minor differences in which LR and SVM might be closer to CV but it's within a .0020 of accuracy which is so minor it makes no difference

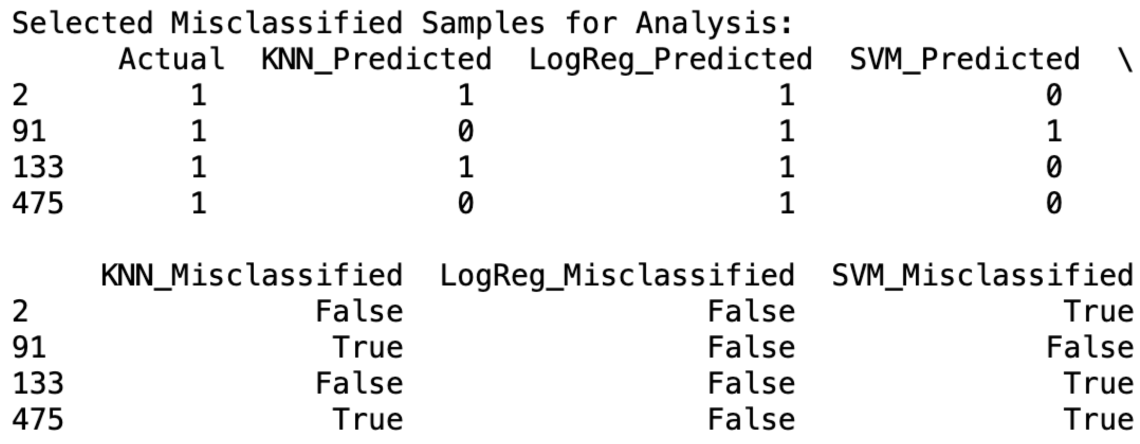
Final Model

All three models performed well, however, SVM (with C parameter) and KNN (With K parameters) are faster compared to LR when upping the C or K. LR can be fast if C is 1 or lower. KNN and SVM might be able to handle large amounts of data frames faster. On training time, LR is slower, but its testing time was faster than the other two. I believe maybe the algorithm to run LR might take more time to get though k-folds since its train and test time was a lot faster compared to the cross validation scores.

I believe that KNN and SVM are the best models so far for my dataset due to the time and accuracy being very reliable on this dataset. I, however, will select SVM as my best model since its accuracy scores are slightly higher and run times very comparable.

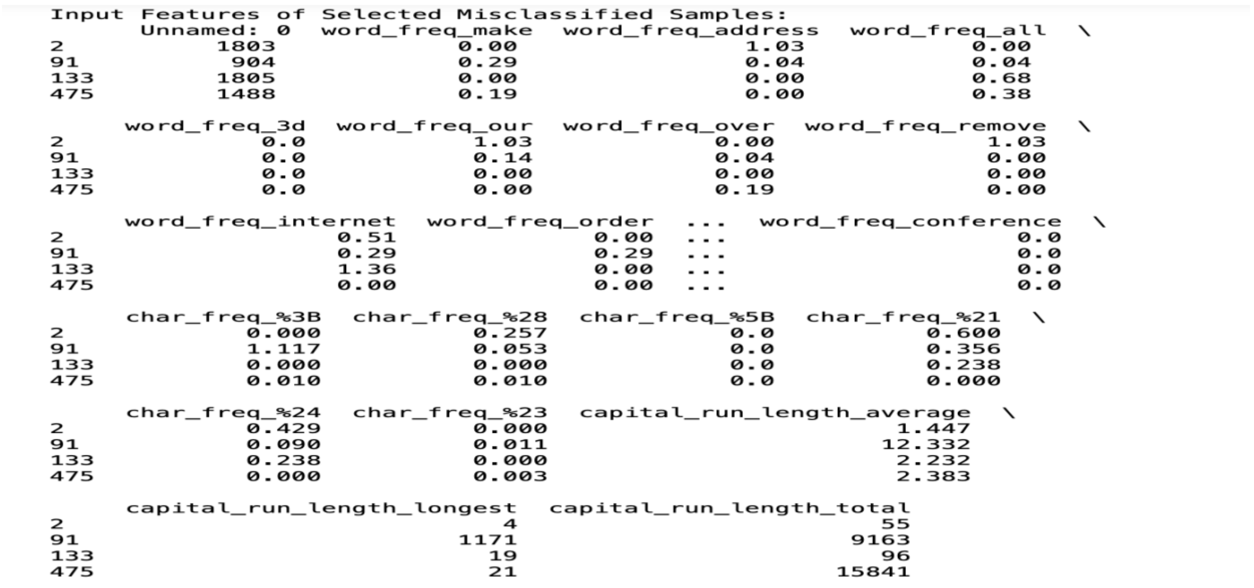
I wanted to analysis my correct outputs with a wrong output to see how some of my models predicted wrong. When running code to check for wrong output vs target output. I got an index that showed this.

Figure 13



```
Selected Misclassified Samples for Analysis:
      Actual  KNN_Predicted  LogReg_Predicted  SVM_Predicted  \
2          1              1                 1              0
91         1              0                 1              1
133        1              1                 1              0
475        1              0                 1              0

     KNN_Misclassified  LogReg_Misclassified  SVM_Misclassified
2                False                 False               True
91                True                 False              False
133              False                 False               True
475               True                 False               True
```

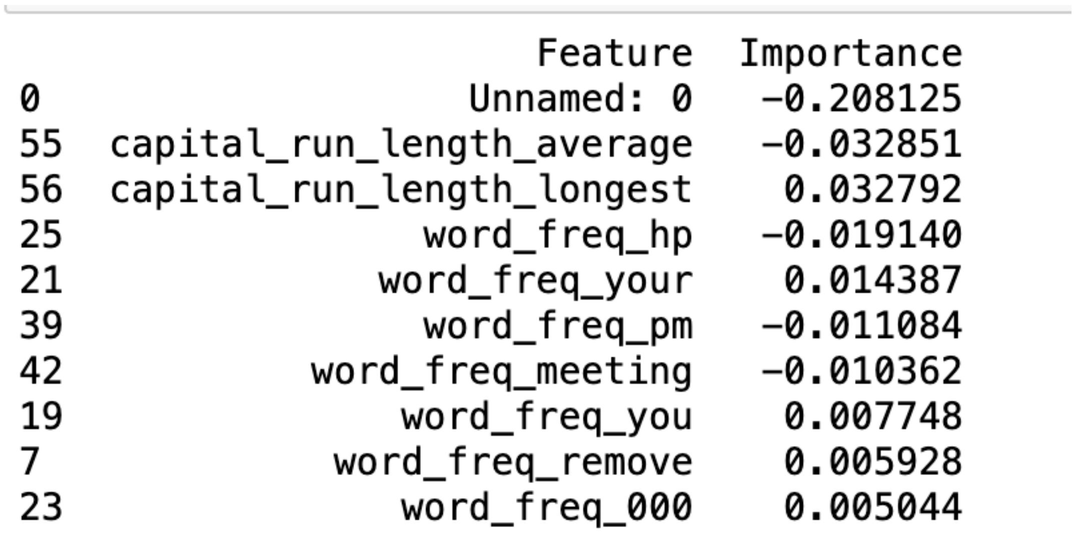
Since for example of these 4 indexes, LR predicted right every time I will be checking its attributes compared to the other two models who might have guessed wrong. Figure 14 shows those indexes instances when those 3 models were trying to predict the correct target output.
Figure 14



```
Input Features of Selected Misclassified Samples:
     Unnamed: 0  word_freq_make  word_freq_address  word_freq_all  \
2          1803            0.00               1.03           0.00
91          904            0.29               0.04           0.04
133        1805            0.00               0.00           0.68
475        1488            0.19               0.00           0.38

     word_freq_3d  word_freq_our  word_freq_over  word_freq_remove  \
2             0.0           1.03            0.00              1.03
91            0.0           0.14            0.04              0.00
133           0.0           0.00            0.00              0.00
475           0.0           0.00            0.19              0.00

     word_freq_internet  word_freq_order  ...  word_freq_conference  \
2                  0.51             0.00  ...                   0.0
91                 0.29             0.29  ...                   0.0
133                1.36             0.00  ...                   0.0
475                0.00             0.00  ...                   0.0

     char_freq_%3B  char_freq_%28  char_freq_%5B  char_freq_%21  \
2            0.000          0.257            0.0          0.600
91           1.117          0.053            0.0          0.356
133          0.000          0.000            0.0          0.238
475          0.010          0.010            0.0          0.000

     char_freq_%24  char_freq_%23  capital_run_length_average  \
2            0.429          0.000                       1.447
91           0.090          0.011                      12.332
133          0.238          0.000                       2.232
475          0.000          0.003                       2.383

     capital_run_length_longest  capital_run_length_total
2                             4                        55
91                         1171                      9163
133                          19                        96
475                          21                     15841
```

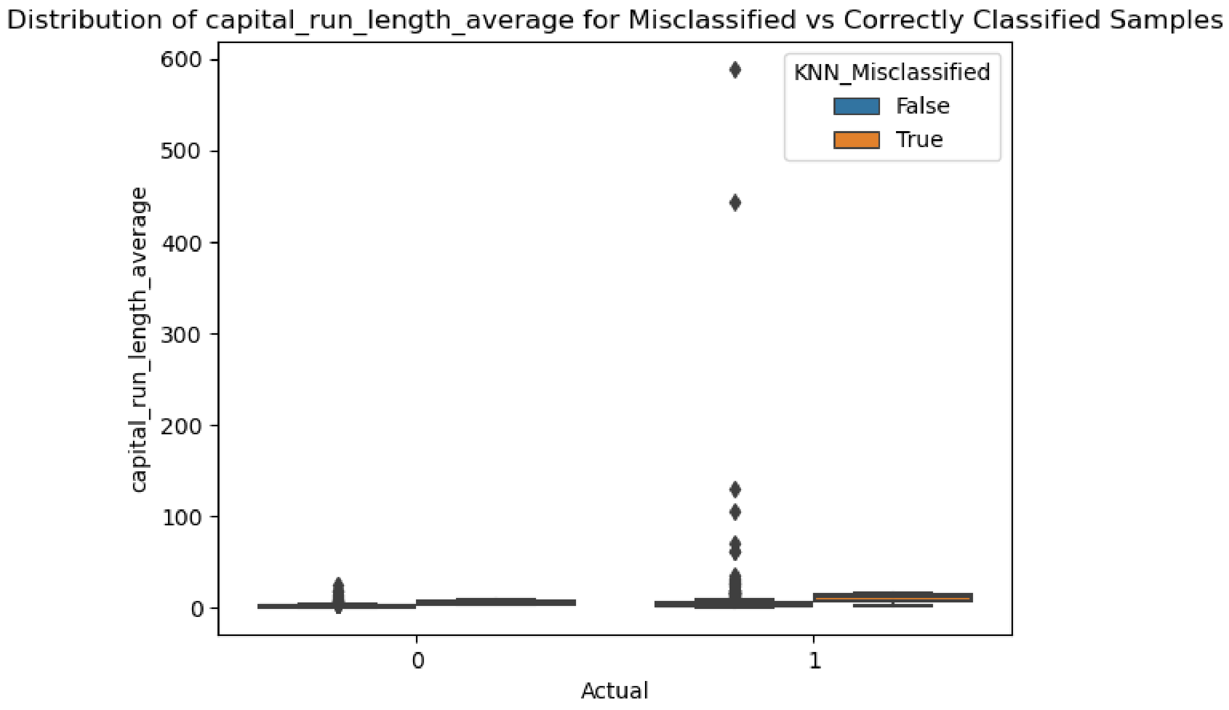
I will be checking why Log gets predictions more correctly and if it has to do with its coefficients on its features. Figure 15 shows LR top ten features based in order top to bottom.

Figure 15



```
                        Feature  Importance
0                   Unnamed: 0   -0.208125
55  capital_run_length_average   -0.032851
56  capital_run_length_longest    0.032792
25                 word_freq_hp   -0.019140
21               word_freq_your    0.014387
39                 word_freq_pm   -0.011084
42            word_freq_meeting   -0.010362
19                word_freq_you    0.007748
7              word_freq_remove    0.005928
23                word_freq_000    0.005044
```

Figure 16



In Figure 16 blue is false which means knn got prediction right and true which is orange means knn got it wrong. It appears on Figure 16. KNN tends to guess correctly when capital run length average is higher and on lower capital nun length average it can guess wrong for KNN since maybe neighbor classification might tend to pick closer targets that could be wrong.

Video Link: https://youtu.be/Itv8O9EQV9U