CSU44061 Machine Learning Project Report

**Group 48**

Deirbhile Walsh 15315676

Vardan Kaushik 20322932

Shengyuan Liu 16341982

# **Introduction**

Our group project is an application based machine learning model. For this project we will be downloading twitter tweets of 6 US politicians. Then, based on these tweets, the model predicts the gender of those politicians. If successful, we will then attempt to predict the exact politician. This application will use multiple machine learning models to train the data and find out which algorithm is most suitable for predicting.

The input to our algorithm are tweets of 6 US politicians. We then use a Logistic Regression, K-Fold Cross-validation, and SVM to output a predicted gender of these politicians.

# **Dataset & Features**

We used the website <https://www.vicinitas.io/> to gather our data. We managed to download roughly 3,000 tweets for each of the politicians, which included both tweets and retweets, and compiled them into a .csv file to be preprocessed.

As our project is setting out to make predictions based on the tweets of specific users, we decided to remove all “retweets” from the dataset as this would negatively impact our predictions. After removing all retweets we were left with 15290 data points between our six politicians.

Following this, we removed all other unnecessary data features such as “tweet ID” and “language”, which we felt wouldn’t help in making predictions. We then added the “gender” column to the dataset as that’s what our model is setting out to predict, leaving us with a dataset containing three features: *Politician Name, Tweet Text* & *Gender.*

Further to this we removed all punctuation and stop words from our tweet text, as they would negatively affect our predictions.

# **Methods Used**

1. Logistic Regression: Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

For Logistic Regression, the model often takes two inputs which one is X the event feature and the other one is Y the result. So Y always has two types, Y=1 or Y=-1 to represent whether the result is positive or negative, X are just features of different events. Model collects data and calculates a decision boundary and sets up itself, when there is a new input X, the model compares it with the boundary to predict if the result will be positive or negative.

1. K-Fold Cross-validation: In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining k−1 subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data. The k results can then be averaged to produce a single estimation.
2. KNN Classifier : K-Nearest Neighbors is a non-parametric method proposed by Thomas Cover used for classification and regression. In KNN Classifier, the input consists of the k closest training examples in the feature space. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors(k is a positive integer, typically small). If k=1, then the object is simply assigned to the class of that single nearest neighbor and by the k increase, the object will be assigned to more neighbours.
3. SVM: A support vector machine(SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. An SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

# **Experiments/Results/Discussions**

* For this project the logistic regression algorithm gave us an accuracy of 84% for predicting gender, and our SVM came closest to that with a very similar accuracy of 83%, which is understandable given that the data is linearly separable and both algorithms utilise very similar loss functions. Both methods also performed similarly in predicting politicians with respective accuracies of 77% and 78%.
* It’s worth mentioning that we didn’t have a perfect split in our dataset, with roughly 8,800 tweets being classed as male and only about 6400 as female. As 15290 tweets was a smaller dataset than we originally hoped for this project, we decided that cutting our data by another 2,000 tweets to get an even split would be too significant a data-loss and would not be worth the trade-off.

# For predicting Gender

## Cross-validation using KFold

### *Cross-validation of CountVector parameters*

In case of choosing the parameters for Countvectorizer, we used cross-validation to choose the best option for the following -

* Max\_features
* Min\_df
* Max\_df
* Ngram\_range
* For cross-validation we used K-Fold cross-validation with a total number of splits equal to 5. We then plotted error-bar for different values of the hyperparameter against the accuracy score of the model trained.

We plotted the errorbar for parameters against the mean accuracy score of the model during cross-validation.

Accuracy score is calculated using the formula - Correct predictions / Total number of Datapoints.

**Error-bars results of cross-validation for max-df**

For cross-validation of max\_df, we took the range of [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1].

**max\_df** is used to remove words that occur in too many documents, words such as “the”, “is”, “a”, that wouldn’t help in predicting the gender.

As we can see from the below plots, the best model to train our data with is logistic regression with the best value for max\_df = 0.6. This means, for best accuracy score, any word with frequency greater than 0.6 will be ignored.

|  |
| --- |
| For Logistic regression, the best value of max\_df is 0.6 with the accuracy score of 0.84.  We can also see a sharp decrease in the accuracy score when no terms are ignored, i.e. for max\_df = 1.  For Logistic Regression : Max\_df  best Max\_df value = 0.6  with accuracy score = 0.8404185742315239 |
| For KNN, the best value of max\_df is 0.4 with the accuracy score of 0.66. Note the accuracy score KNN model is significantly low and in this plot too, a sharp decrease in the accuracy score for max\_df = 1.  For KNN : Max\_df  best Max\_df value = 0.4  with accuracy score = 0.6674296926095488 |
| For Support Vector Classifier , the best value of max\_df is 0.6 with the accuracy score of 0.83 which is close to that of logistic regression however SVC doesn’t converge even after 2000 iterations. There is a sharp decrease in the accuracy score in this plot too.  For SVC : Max\_df  best Max\_df value = 0.6  with accuracy score = 0.8376716808371484 |

**Error-bars results of cross-validation for min\_df**

For cross validation of min\_df we took a range of [0, 0.1, 0.2, 0.3, 0.4]

|  |
| --- |
| For Logistic regression, the best value of min\_df is 0 with the accuracy score of 0.83.  For Logistic Regression : Min\_df  best Min\_df value = 0  with accuracy score = 0.8396337475474166 |
| For KNN, the best value of max\_df is also 0 with the accuracy score of 0.64. Note : Accuracy score KNN model is significantly low.  For KNN : Min\_df  best Min\_df value = 0  with accuracy score = 0.6482014388489208 |
| For Support Vector Classifier , the best value of max\_df is also 0 with the accuracy score of 0.83 which is close to that of logistic regression however SVC doesn’t converge even after 2000 iterations.  For SVC : Min\_df  best Min\_df value = 0  with accuracy score = 0.8364944408109876 |

As we can see from the above plots, the best value for min\_df is 0 for all the models. This means that any word with frequency smaller than 0 will be ignored, which is no word. It can also be observed that the best model to train this data with is Logistic Regression.

**Error-bars results of cross-validation for max\_features**

The range of max\_features taken for cross-validation is - [100, 500, 1000, 2000, 3000, 5000].

|  |
| --- |
| For Logistic regression, the best value of Max\_features is 5000 with the accuracy score of 0.83.  For Logistic Regression : max\_features  best Max\_features value = 5000  with accuracy score = 0.8396337475474166 |
| For KNN, the best value of max\_features is 500 with the accuracy score of 0.67. Note : Accuracy score KNN model is significantly low.  For KNN : max\_features  best Max\_features value = 500  with accuracy score = 0.6737737083060825 |
| For SVC, the best value of max\_features is also 5000 with the accuracy score of 0.83 which is close to that of logistic regression however SVC doesn’t converge even after 2000 iterations.  For SVC : max\_features  best Max\_features value = 5000  with accuracy score = 0.8364944408109876 |

The best value of max\_features is 5000 for both Logistic regression as well as Support Vector Classifier, however, for the KNN classifier the best value of max\_features is 500.

As we can see, even at the best value of max\_feaures, for KNN model, the accuracy score of the prediction remains close to 0.67 which is pretty low than the accuracy score of SVC and Logistic Regression, which is close to 0.84. Therefore, we will be using 5000 for max\_features to get maximum accuracy score as possible.

We didn’t use any value of the max\_features greater than 5000 as we didn’t want features to exceed 5000 limit. 5000 features already make a huge dataset, for any value greater than 5000, model training would have taken a tremendous amount of time and RAM, which we didn’t want.

**Error-bars results of cross-validation for the maximum value of ngram\_range**

For ngram\_range, the minimum value we kept fixed, i.e. 1, and we used cross-validation for the maximum value of the ngram\_range.

We took values for max ngram\_range as - [1, 2, 3]. As pairing range increases, the size of the vocabulary also increases, so we didn’t take any value greater than 3.

|  |
| --- |
| For Logistic regression, the best value of max ngram\_range is 2 with the accuracy score of 0.81.  For Logistic Regression : ngram\_range  best ngram value = 2  with accuracy score = 0.8194898626553304 |
| For KNN, the best value of max ngram\_range is 2 with the accuracy score of 0.67. Note : Accuracy score KNN model is significantly low.  For KNN : ngram\_range  best ngram value = 3  with accuracy score = 0.6626553302812296 |
| For Support Vector Classifier , the best value of maximum ngram\_range is also 2 with the accuracy score of 0.81 which is close to that of logistic regression however SVC doesn’t converge even after 2000 iterations.  For SVC : ngram\_range  best ngram value = 2  with accuracy score = 0.816219751471550 |

We only used cross validation for the maximum value of the ngram\_range and found that (1,2) is the best value of the ngram\_range. Since SVC didn’t converge and KNN accuracy score is pretty low, so again Logistic regression is the best model to train this data with.

In a nutshell, for CountVectorization, best values for -

* Max\_df = 0.6
* Min\_df = 0
* Max\_features = 5000
* ngram\_range = (1,2)

Moreover, the best model to train the data with is Logistic Regression.

### 

### *Cross-validation of model parameters*

Now, we move onto cross-validating parameters of

* Logistic regression
* KNN classifier
* SVC

**Error-bars results of cross-validation for the model parameter**

|  |
| --- |
| The plot in the right is the maximized version of the plot in the left. We took C range as [0.01, 0.1, 1, 10, 100]. We see that the maximum accuracy, 0.84, is achieved at C = 1 for logistic Regression.  For Logistic Regression  best value of C - 1  with accuracy = 0.8404185742315239 |
| Range of K neighbors or n\_neighbors parameters used for cross validation is - [2, 3, 4, 5, 6, 7, 8, 9, 10]. We see a gradual decrease in the accuracy score as the n\_neighbors increases. Best value of n\_neighbors for KNN is 2 with accuracy score = 0.43. This is worse than the baseline model which is discussed further in the report. So KNN should not be used as model training.  For KNN  best value of K - 2  with accuracy = 0.4387181164159582 |
| For SVC, the best value of C parameter is 0.1. With C = 0.1, accuracy of SVC model is equal to 0.83 which is pretty close to that of logistic regression. As seen in the plot, as the value of C increases, accuracy score decreases. As C is a regularization parameter and regularization is inversely proportional to C value, so the decreasing accuracy score makes sense as the number of features are huge in this dataset, ranging towards 5000.  Note : for C = 10, the SVC model doesn’t converge even with max\_iter = 2000.  For SVC  best value of C - 0.1  with accuracy = 0.8376716808371484 |

In a nutshell, the best model to use is Logistic Regression with C = 1.

## **Choosing best model**

For choosing the best model, we used confusion matrices, accuracy scores and plotted ROC curves.

### *Confusion matrix*

Confusion matrix is given as -

|  |  |
| --- | --- |
| True Negative | False Positive |
| False Negative | True Positive |

Greater the values of True Positive and True negative, better our model trained as better will be the accuracy of the model.

For Logistic Regression -

Confusion matrix for Logistic Regression model -

[[1024 297]

[ 184 1553]]

Correct predictions = 2557, False predictions = 481, Total number of datapoints = 3058.

Accuracy score = 2557/3058 = **0.83** which is consistent with the accuracy score results found during cross validation.

For KNN classifier -

Confusion matrix for KNN model -

[[ 740 581]

[ 524 1213]]

Correct predictions = 1953, False predictions = 1105, Total number of datapoints = 3058.

Accuracy score = 1953/3058 = **0.63** which is consistent with the accuracy score results found during cross validation.

For SVC classifier -

Confusion matrix for SVC model -

[[1021 300]

[ 190 1547]]

Correct predictions = 2568, False predictions = 490, Total number of datapoints = 3058.

Accuracy score = 2568/3058 = **0.83** which is consistent with the accuracy score results found during cross validation.

For dummy classifier -

Confusion matrix for Dummy classifier -

[[ 0 1321]

[ 0 1737]]

Notice no negative (female in this case) predictions have been made. This is because the dummy classifier predicts the most frequent class and as said in our dataset features section, we have more male classes than females.

Correct predictions = 1737, False predictions = 1321, Total number of datapoints = 3058.

Accuracy score = 1737/3058 = **0.56**.

|  |  |
| --- | --- |
| **Model** | **Accuracy Score** |
| Logistic Regression | 0.83 |
| KNN | 0.63 |
| SVC | 0.83 |
| Dummy classifier | 0.56 |

So the best models to choose from are Logistic Regression and Support Vector Classifier.

### *ROC curves*

In ROC curves we look for the point with highest True positive rate with the lowest false positive rate. This means that the top left corner of the plot is the “ideal” point - a false positive rate of zero, and a true positive rate of one. The red-line in the middle of the plot signifies our baseline model, so our trained model is better than the baseline model if its ROC curve is above the line.

Logistic Regression and SVM have almost identical performance as seen below. Area under the curve(AUC) is significantly lower for K-neighbours, as expected.

|  |
| --- |
| The best value from the plot seems to be around 0.2 of False positive rate and 0.8 of True Positive rate. No part of the ROC curve is below the red line and the curve seems to enclose a large enough area under it, which is a good thing. |
| There is no steep rise in the ROC curve for KNN, also, the curve is pretty close to the red line, which is not a good thing for a training model. Not to mention, it doesn’t enclose large area under it, so we wouldn’t use KNN for training this data. |
| Just like in the case of Logistic Regression, the best value from the plot seems to be around 0.2 of False positive rate and 0.8 of True Positive rate. Moreover, no part of the ROC curve is below the red line and the curve seems to enclose a large enough area under it, which is a good thing. |

## Model Training

From the results we obtained from cross validation, we will be using Logistic Regression, (with C = 1) and CountVectorization with ( max\_features = 5000, min\_df = 0, max\_df = 0.6 & ngram\_range = (1,2)).

Dividing our dataset into test, training data using -

**from sklearn.model\_selection import train\_test\_split**

**X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_features,y\_gender,test\_size=0.2)**

80% of the dataset will be used for training and 20% for testing.

Training Logistic Regression model -

from sklearn.linear\_model import LogisticRegression

LR\_model = LogisticRegression(C=1, max\_iter=2000)

LR\_model.fit(X\_train,y\_train)

LR\_predict = LR\_model.predict(X\_test)

print("Logistic Regression : {}".format(accuracy\_score(y\_test, LR\_predict)))

This gives the output -

Logistic Regression accuracy score : 0.841399607586658.

**Baseline model comparison**

So we got 0.84 as an accuracy score, but is it better than a baseline model?

We trained a dummy classifier predicting the model frequent class as -

from sklearn.dummy import DummyClassifier

dummy = DummyClassifier(strategy="most\_frequent")

dummy.fit(X\_train,y\_train)

dummy\_prediction = dummy.predict(X\_test)

print("Dummy classifier accuracy score : {}".format(accuracy\_score(y\_test,dummy\_prediction)))

For the baseline model predicting the model frequent class, the accuracy score is 0.56.

Dummy classifier accuracy score : 0.5601177240026161

So, definitely our Logistic regression model is better than the baseline model as 0.84 is greater than 0.56.

***Did we over-fit?***

To check if we over-fit, we calculated the accuracy score on the training dataset. If this turns out to be close to 100, then yes indeed we over-fitted our dataset.

To calculate accuracy score on the training dataset -

LR\_model.fit(X\_train,y\_train)

LR\_predict = LR\_model.predict(X\_train)

print("Logistic Regression : {}".format(accuracy\_score(y\_train, LR\_predict)))

From this we get the accuracy score of 0.94.

Logistic Regression training accuracy : 0.9436723348593852

Considering the fact that the training accuracy score is nowhere near 100 along with the fact that the bars in the error bars plots (signifying variance in the accuracy of different splits) of logistic regression are not big in size, we can confidently say that we did not overfit the data.

We also used the best possible parameter values of all the hyperparameters used in this code, validated using cross validation, therefore we also did not underfit our data.

# For predicting Politician Name

## *Cross-validation using KFold*

Cross validation for count vectorization yields the same results, so we will just include the Error-Bars for cross validation of models.

### *Cross-validation for parameters of training models* -

**Error-bars results of cross-validation for the model parameter**

|  |
| --- |
| Again we took C range as [0.01, 0.1, 1, 10, 100], giving us a max accuracy of 0.77 with C=1.  For Logistic Regression  best value of C - 1  with accuracy = 0.772661870503597 |

|  |
| --- |
| Again using [2, 3, 4, 5, 6, 7, 8, 9, 10] as our range of K neighbours, we achieved a max accuracy of 0.44 when K=2. Once again, KNN is not sufficient.  For KNN  best value of K - 2  with accuracy = 0.4387181164159582 |

|  |
| --- |
| Again we took C range as [0.01, 0.1, 1, 10, 100], with our best overall accuracy for politician prediction at 0.78, using a value of C=0.1.  For SVC  best value of C - 0.1  with accuracy = 0.780313930673643 |

Clearly, the Logistic Regression model and Support Vector classifier are the best models to predict politician names. Logistic regression with C value equal to 1 and SVC with C value equal to 0.1.

## Choosing best model

For choosing the best model, we used confusion matrices and accuracy scores. We didn’t plot ROC curves in this case as ROC are mainly used for binary classification and predicting politicians is a multiple classification problem.

### Confusion matrix

Notice the 6x6 confusion matrix, this is because now we have 6 classes for prediction.

True predictions are still given by the sun of the diagonal elements.

For Logistic Regression -

Confusion matrix for Logistic Regression model -

[[466 14 38 9 21 17]

[ 4 569 21 14 11 2]

[ 27 49 321 42 70 5]

[ 15 32 44 421 58 1]

[ 15 16 62 58 380 6]

[ 14 10 13 6 11 196]]

Correct predictions = 2353, False predictions = 705, Total number of datapoints = 3058.

Accuracy score = 2353/3058 = **0.76** which is consistent with the accuracy score results found during cross validation.

For KNN classifier -

Confusion matrix for KNN model -

[[449 88 22 4 1 1]

[150 445 20 6 0 0]

[165 246 86 12 5 0]

[120 181 80 176 14 0]

[168 175 87 52 55 0]

[ 97 76 20 6 4 47]]

Correct predictions = 1258, False predictions = 1800, Total number of datapoints = 3058.

Accuracy score = 1258/3058 = **0.41** which is consistent with the accuracy score results found during cross validation.

For SVC classifier -

Confusion matrix for SVC model -

[[483 10 30 8 22 12]

[ 2 582 13 13 10 1]

[ 25 59 325 35 63 7]

[ 13 31 37 436 53 1]

[ 17 18 51 60 389 2]

[ 14 13 12 3 8 200]]

Correct predictions = 2415, False predictions = 643, Total number of datapoints = 3058.

Accuracy score = 2415/3058 = **0.78** which is consistent with the accuracy score results found during cross validation.

For dummy classifier -

Confusion matrix for Dummy classifier -

[[ 0 0 0 565 0 0]

[ 0 0 0 621 0 0]

[ 0 0 0 514 0 0]

[ 0 0 0 571 0 0]

[ 0 0 0 537 0 0]

[ 0 0 0 250 0 0]]

Notice only one column of the matrix had non zero elements, rest all of the elements are equal to zero. This is because the dummy classifier predicts the most frequent class and this time output classes are the names of the politician mentioned in the Dataset & Feature section. So, each column for each politician and dummy classifier always predicts the most frequent tweeter, most probably Donal Trump.

Correct predictions = 571, False predictions = 2487, Total number of datapoints = 3058.

Accuracy score = 571/3058 = **0.18.**

|  |  |
| --- | --- |
| **Model** | **Accuracy Score** |
| Logistic Regression | 0.76 |
| KNN | 0.41 |
| SVC | 0.78 |
| Dummy classifier | 0.18 |

So the best model to choose is the Support Vector Classifier.

Model Training

From the results we obtained from cross validation, we will be using Logistic Regression, (with C = 1) and CountVectorization with ( max\_features = 5000, min\_df = 0, max\_df = 0.6 & ngram\_range = (1,2)).

Dividing our dataset into test, training data using -

**from sklearn.model\_selection import train\_test\_split**

**X\_train,X\_test,y\_train,y\_test = train\_test\_split(X\_features,y\_person,test\_size=0.2)**

80% of the dataset will be used for training and 20% for testing.

Training SVC model -

from sklearn.svm import LinearSVC

SVC\_model = LinearSVC(C=0.1,max\_iter=2000)

SVC\_model.fit(X\_train,y\_train)

SVC\_predict = SVC\_model.predict(X\_test)

print("SVC accuracy score : {}".format(accuracy\_score(y\_test,SVC\_predict)))

This gives the output -

SVC accuracy score : 0.78973185088293

**Baseline model comparison**

So we got 0.789 as an accuracy score, but is it better than a baseline model?

We trained a dummy classifier predicting the model frequent class as -

from sklearn.dummy import DummyClassifier

dummy = DummyClassifier(strategy="most\_frequent")

dummy.fit(X\_train,y\_train)

dummy\_prediction = dummy.predict(X\_test)

print("Dummy classifier accuracy score : {}".format(accuracy\_score(y\_test,dummy\_prediction)))

For the baseline model predicting the model frequent class, the accuracy score is 0.18.

Dummy classifier accuracy score : 0.1867233485938522

So, SVC is definitely much better than the baseline model as 0.79 is much greater than 0.18.

**Did we over-fit?**

To check if we over-fit, we calculated the accuracy score on the training dataset. If this turns out to be close to 100, then yes indeed we over-fitted our dataset.

To calculate accuracy score on the training dataset -

SVC\_model.fit(X\_train,y\_train)

SVC\_predict = LR\_model.predict(X\_train)

print("SVC training Accuracy : {}".format(accuracy\_score(y\_train, SVC\_predict)))

From this we get the accuracy score of 0.93.

SVC training accuracy : 0.9355788096795291

Considering the fact that the training accuracy score is nowhere near 100 along with the fact that the bars in the error bars plots (signifying variance in the accuracy of different splits) of logistic regression are not big in size (meaning smaller variance in accuracy score of different splits), we can confidently say that we did not overfit the data.

We also used the best possible parameter values of all the hyperparameters used in this code, validated using cross validation, therefore we also did not underfit our data.

# **Summary**

In general, we sorted the data from the website into different big tables to represent different features of the data. Then we used several machine learning algorithms which we had learned this term to train these politicians’ tweets, predict their gender and compare results with real data. We are satisfied with the prediction accuracy that can be achieved by our application. After all, the result of both the Logistic Regression and SVM has a high accuracy of the model to predict the gender of politicians by inputting their tweets. Which the accuracy could reach over 83%, which also exceeds our basic target accuracy which is 80%. For the SVM we realize that because this algorithm is used for the text and female politicians’ are often use more academic words and more technical terms on their twitter but the male politicians’ tweets are more colloquial and shorter so its accuracy is higher. For Logistic Regression, since female politicians tend to tweet longer and more logically, this gives the algorithm more distinct features in construction and training.It also makes the difference between male and female politicians more obvious, and the decision boundaries of the model are clearer, so the prediction accuracy of the model is very high. As for KNN and K-Fold Cross-validation, since they are not the right algorithms for this situation, their predictions’ accuracy are all not so high and can not reach our standard. Given that we had the time, we attempted to predict the exact politician, which was most accurate using an SVC with C value of 0.1, giving us an accuracy of 78%.

# **Contributions**

Vardan Kaushik: Created and made the main function code, added the Logistic Regression function, added the SVM, wrote the Result part of report, and wrote the project proposal. Also responsible for the code to predict the politician.

Shengyuan Liu : Wrote Introduction, Methods, Summary and Contributions parts of report and added KNN Classifier functions and drew graph functions to application code.

Deirbhile Walsh : Divide & assign the project, wrote Dataset & Features, Experiments and Discussion parts of report, collected & preprocessed data and added the K-Fold cross validation functions.

# **Project Link:**

**Github link:** [**https://github.com/VardanKaushik/Machine-Learning-Group-Project**](https://github.com/VardanKaushik/Machine-Learning-Group-Project)