Intelligent Systems Engineering

Proceedings Spring 2019

G. C. Fox

G. von Laszewski

T. Balson

Editor

laszewski@gmail.com

https://github.com/cloudmesh-community/proceedings

May 03, 2019 - 11:30 AM

Created by Cloudmesh & Cyberaide Bookmanager, https://github.com/cyberaide/bookmanager

Contents

1 PREFACE 5

1.1 Disclaimer . . . 5

1.2 Contributors . . . 5

1.3 Editor . . . 5

1.4 abstract . . . 6

2 S-cone Classification Using REST Services and Machine Learning 7

2.1 Abstract . . . 7

2.2 Introduction . . . 7

2.3 Data . . . 9

2.3.1 Preprocessing . . . 10

2.3.2 Visualization . . . 10

2.4 Model Discussion . . . 10

2.4.1 Failures . . . 11

2.4.2 Activation Function . . . 12

2.4.3 Decision . . . 13

2.5 REST Service Implementations . . . 13

2.6 Specification . . . 14

3 Rookie Fantasy Football Point Prediction 16

3.1 Abstract . . . 16

3.2 Introduction . . . 16

3.3 Data Set . . . 17

3.4 KNN Algorithm . . . 17

3.5 Implementation . . . 19

3.6 Limitations . . . 21

3.7 Conclusion . . . 21

3.8 Specification . . . 22

4 Analysis of soccer data with kmeans 24

4.1 Abstract . . . 24

4.2 Introduction . . . 24

4.2.1 Sport Analytics . . . 25

4.2.2 Sensors in Sports . . . 25

4.3 Soccer Dataset . . . 26

4.4 Algorithm Discussion . . . 27
4.4.1 K-means . . . 27
4.4.2 Spectral Clustering . . . 28
4.4.3 Dimensional Reduction . . . 28
4.5 Results . . . 29
4.6 Analysis . . . 33
4.7 Specification . . . 34

5 Tetris Score Analysis Server 37
5.1 Abstract . . . 37
5.2 Introduction . . . 37
5.3 Design . . . 38
5.4 Architecture . . . 38
5.5 Dataset . . . 39
5.6 Results . . . 39
5.7 Conclusion . . . 42
5.8 Specification . . . 42

6 Political Bias and Voting Trends 45
6.1 Abstract . . . 45
6.2 Introduction . . . 45
6.3 Requirements . . . 45
6.4 Design . . . 46
6.4.1 Python . . . 46
6.4.2 REST Service . . . 46
6.4.3 Docker . . . 47
6.5 Dataset . . . 47
6.6 Results . . . 47
6.7 Discussion . . . 48
6.8 Conclusion . . . 49
6.9 Work Breakdown . . . 49
6.10 Specification . . . 50

7 Spam Analysis with Spamalot 53
7.1 Abstract . . . 53
7.2 Introduction . . . 53
7.3 The Algorithm . . . 54
7.3.1 Naive Bayes . . . 54
Metrics . . . 55

7.3.2 Support Vector Machines (SVM) . . . 55
7.4 The Data Set . . . 57
7.5 Model Results . . . 57
7.6 Implementation . . . 59
7.6.1 The Server . . . 59
7.6.2 The Upload Function and Classification . . . 60
7.6.3 Specification . . . 61
7.7 Conclusion . . . 61

8 REFERENCES 63

1 PREFACE

1.1 Disclaimer

> "This document was produced with Cyberaide Bookmanager developed by Gregor von Laszewski available at https://pypi.python.org/pypi/cyberaide-bookmanager. It is in the responsibility of the user to make sure an author acknowledgement section is included in your document. Copyright verification of content included in a book is responsibility of the book editor."

1.2 Contributors

Hid	Firstname	Lastname	Community	Semester
sp19-222-100	Saxberg	Jarod	e222	sp19
sp19-222-101	Bower	Eric	e222	sp19
sp19-222-102	Danehy	Ryan	e222	sp19
sp19-222-89	Fischer	Brandon	e222	sp19
sp19-222-90	Japundza	Ethan	e222	sp19
sp19-222-91	Zhang	Tyler	e222	sp19
sp19-222-92	Yeagley	Ben	e222	sp19
sp19-222-93	Schwantes	Brian	e222	sp19
sp19-222-94	Gotts	Andrew	e222	sp19
sp19-222-96	Olson	Mercedes	e222	sp19
sp19-222-97	Levy	Zach	e222	sp19
sp19-222-98	McDowell	Xandria	e222	sp19
sp19-222-99	Badillo	Jesus	e222	sp19

1.3 Editor

Many edits to the paper wer performed by Tyler Balson and Gregor von Laszewski. They also made in many cases significant suggestions for improving the quality of the papers.

1.4 abstract

write abstract

2 S-cone Classification Using REST Services and Machine Learning

Brandon Fischer, Ryan Danehy
bfisch9@iu.edu, rdanehy@iu.edu
Indiana University Bloomington
hid: sp19-222-89 sp19-222-102
github: ☒
code: ☒

Keywords: S-cones, Scikit, Support Vect Machine, Neural Network, WebPlotViz, ISOS (Inner Segment - Outer Segment junction), Cone Outer segment tip (COST).

2.1 Abstract

We worked in partnership with Dr. Don Miller's lab from the IU School of Optometry to create a binary classifier which is trained to differentiate (and generate a count of) S-cones from L and M cones in 3D retinal imaging. We deployed an RBF-kernel SVM to classify S-cones vs non-S-cones. This project has clinical significance in the tracking of progression of the disease Retinitis Pigmentosa (RP). In RP, S-cones can be seen migrating from their natural positions, and eventually disappearing entirely in retinal scans. Our service could be extended from purely classifying/counting S-cones to tracking the rate of their movement and determining the progression/severity of the disease in a given patient.

2.2 Introduction

Cones or Cone cells are photoreceptor cells in the retinas of humans. They are responsible for color vison and work best in bright lights. S-cone cells differ from M-cones and L-cones based on the light wavelengths they are senstive to. For example, S-cones are sensitive to short-wavelengths, M-cones to medium-wavelengths, and L-cones to Long-wavelengths 1. Short-wavelengths correspond with "blue", medium with "green", and long with "red", therefore it is believed that the study of these cones could lead to new insights into dieseases such as red-green colorblindness.

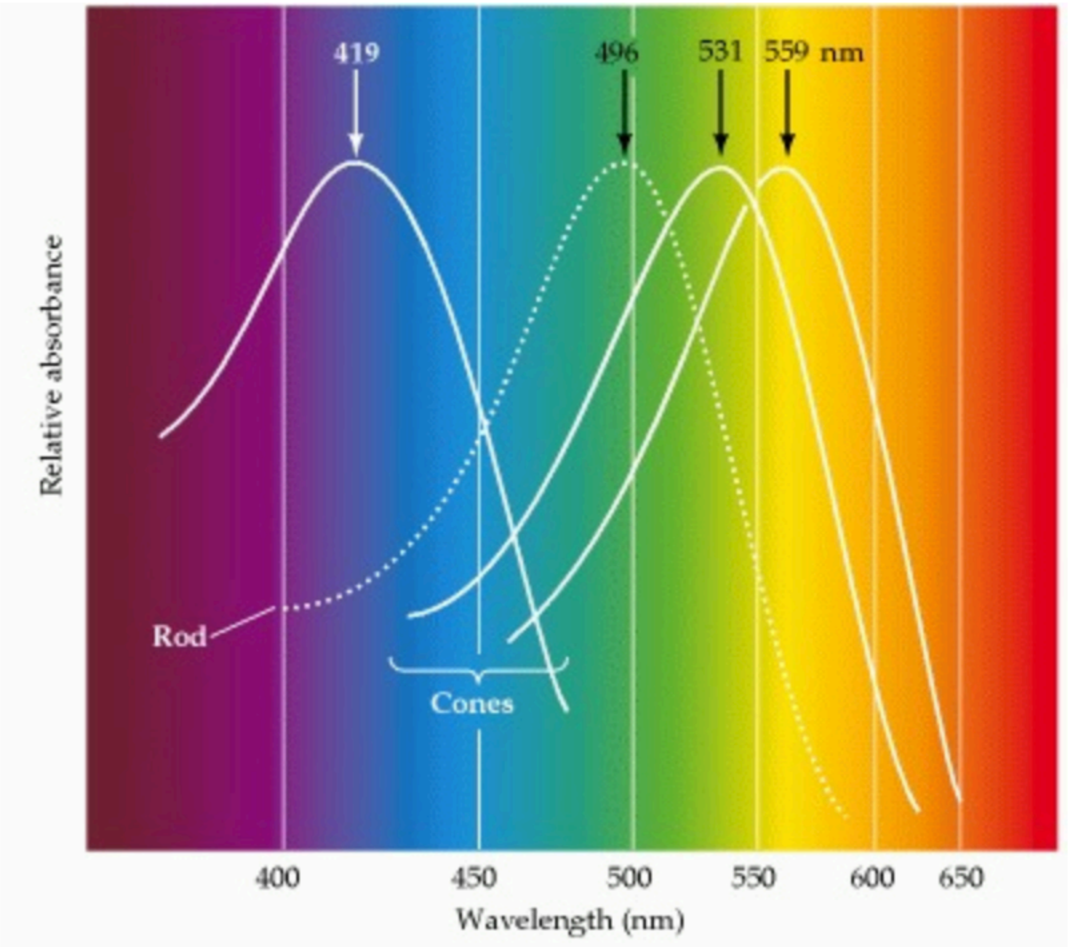


Figure 1: Spectrum of Wavlengths

Caption: Color vision. The absorption spectra of the four photopigments in the normal human retina. The solid curves indicate the three kinds of cone opsins; the dashed curve shows rod rhodopsin for comparison. Absorbance is defined as the log value of the intensity of incident light divided by intensity of transmitted light 1.

Individual cones are entirely color blind in that their response is simply a reflection of the number of photons they capture, regardless of the wavelength of the photon. Therefore it is impossible to determine why a change in the effectiveness of a particular cone occurred. This question can only be resolved by comparing the activity in different classes of cones. Comparisons of the responses of individual cone cells, and cells at higher levels in the visual pathway are clearly involved in how the visual system extracts color information from spectral stimuli. However, understanding of the neural mechanisms that underlie color perception has been elusive to the scientific community Figure 1.

Figure:Cone_mosiac caption:“This diagram was produced based on histological sections from a human eye to determine the density of the cones. The diagram represents an area of about 1° of visual angle. The number of S-cones was set to 7% based on estimates from previous studies. The L-cone:M-cone ratio was set to 1.5. This is a reasonable number considering that recent studies have shown wide ranges of cone ratios in people with normal color vision. In the central fovea an area of approximately 0.34° is S-cone free. The S-cones are semi-regularly distributed and the M- and L-cones are randomly distributed. Throughout the whole retina the ratio of L- and M- cones to S-cones is about 100:1 2." There are two main reflection sites inside the cone photoreceptor cells that line the back of the eye. The first one occurs at what is called the inner segment – outer segment junction (ISOS) and the second one (which occurs directly behind the first one) occurs at what is called the cone outer segment tip (COST). Cones can be classfied by the comparison of the inner segment length vs outer segment length. For example, histologically S-cones have a longer inner segment and a shorter outer segment. The ultimate goal of this project was to design an effective and fast method of classification of S-cones, using REST Services to facilitate user interaction with the model.

2.3 Data

Datasets were provided by Dr. Miller's lab which included the 3D coordinates and aperture size of each cone detected within the retinal scan. Using this information, we were able to differentiate the S-cones from the others due to their deeper position and wider aperture compared to the other cell types. Our starting dataset includes information from the images of three patients' retinas, with a mix of healthy and colorblind individuals. Additional data was collected as needed. The data provided was delivered in a fashion to not jeopadize the patients anonymity nor were we given any personal data that could or would put a patient's privacy in concern.

After the initial model was trained, unlabeled data is given a classification through a rest service, and the count and locations of S-cones was returned via another rest service. The service also allows the retraining of the model on new datasets, and outputs the corresponding metrics on the newly trained model. This allows the model to be updated and re-trained as more data becomes available.

Our data includes eight features: X-Coordinate of cone (Coord_X), Y Coordinate of cone (Coord_Y),Retina Depth location of Inner- Outer Segment(ISOS_Z), X Coordinate of ISOS (ISOS_size_X), Y Coordinate of ISOS (ISOS_size_Y), Retina Depth location of COST (COST_Z), X Coordinate of COST (COST_size_X), and Y Coordinate of COST (COST_size_Y). These features were extracted from 3D imaging of the retinal hence the three dimensional parameter types. ISOS_z is the retinal depth location of ISOS and COST_z is the retinal depth location of COST.

2.3.1 Preprocessing

Our raw data had some observations that were unknown or missing certain datapoints, and as such were marked “Nan” in the original dataset. We preproccessed the data in order to exclude feature vectors which included Nan for any feature value. The preproccessing that we performed on the data can be seen in the read_data.py file. Machine learning models can be very sensitive to scaling and in order to prevent this we normalized the data. Normalization rescales the data to be in the range of 0 to 1, thus eliminating any possible feature scaling within our data. We performed normalization on our data using Scikit learn’s preprocessing.normalize() function. This function scales the input indivudally to unit norms (vector length). In order to guarantee the quality of our data before training a model, we standardized our data. Standardization transforms data to have a mean of zero and standard deviation of 1. Standardization was performed by using scikit learn’s StandardScaler(). The function first_model and retrain_model in model.py show this normalization and standardization.

2.3.2 Visualization

We visualized our data using WebPlotViz which results can be seen using the following link. https://spidal-gw.dsc.soic.indiana.edu/dashboard. From the WebPlotViz visualizations it can be noted that the data is not clearly seperated into clusters nor in a regular shape. It also imporant to notice how there is no clear distiniction on which features are weighted heaviler than others in classifying S-cone from M and L-cones. However, histological studies have shown that the biggest differentiation betwen the different cones types is the difference between ISOS_Z - COST_Z 3. This difference siginifes the physical length of an important component of the cone photoreceptors. In one of the visualizations we plotted X_coordinate vs Y-coordinate vs (COST_Z- ISOS_Z). In this plot it is not glaringly obvious that (COST_Z- ISOS_Z) is the most imporant feature, but there does seem to be a noticeable correlation. The lack of an obviously dominant feature led us to the conclusion that for our model to train the best no weights should be applied (*Not sure that’s accurate, I think DNN will always give weights*)

2.4 Model Discussion

In our final project we decided to use a neural network using scikit learn’s MLPClassifier(). We had several reason for picking a neural network model, one being the fact that it is supervised learning. Supervised learning make sense for our project because in our training data we were given labels for every cell observed. Supervised learning also made sense given that our goal was to predict the type of cone for a given cell, and making predictions is usually the goal behind supervised learning algorithms. Additionally, neural networks are ideal for solving non-linear classification problems, which is precisely what we are trying to solve [4].

2.4.1 Failures

We started out trying to use a support vector machine algorithm (SVM) as our model of choice, but several problems became evident while trying to implement the SVM. We originally choose Scikit's svm.SVC() algorithm, but after testing it became apparent the model was not accurate at all averaging an F1 score below .3. We then experimented with altering the parameters of the algorithm, including changing from a linear to non-linear SVM model. We found that changing the kernel to "RBF" produced an F1 score of 1.0. This makes sense given that our data is grouped in a nonlinear way and "RBF" are used for nonlinear solutions.

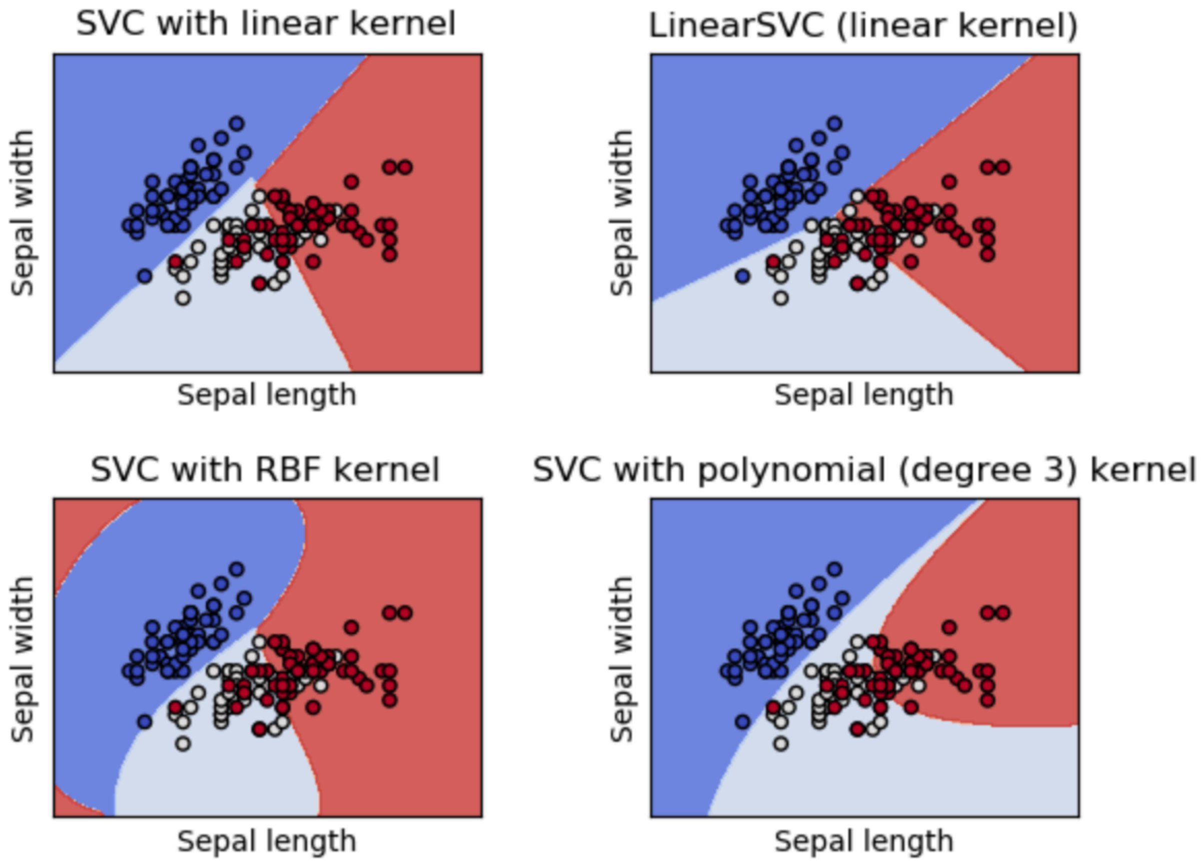


Figure 2: RBF Kernel example

However, an F1 score of 1.0 is a possible symptom of an over-fitting problem. Over-fitting is when an algorithm matches so perfectly to the training data that the model does not generalize well to classification of additional datasets.

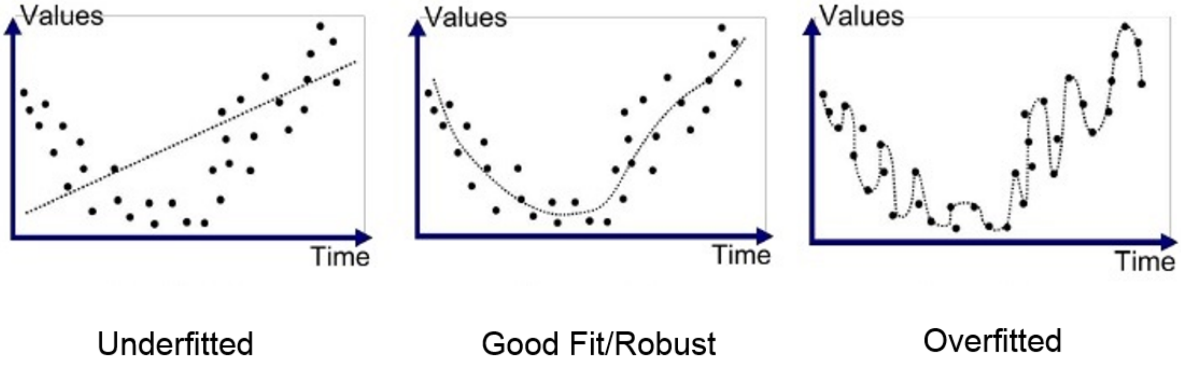


Figure 3: Overfitting example

To determine whether there is overfitting or not we decided to decrease the amount of data used for training. Decreasing the amount of data used for training should only slightly decrease the performance of the model unless there is overfitting. If there was overfitting then they would be a drastic difference since they would be less data to fit perfectly too. We trained with roughly 60% of the data rather than 80%. The new model that was trained with 60% of the data performed horribly, confirming the overfitting problem. Thus we decided to move on to the implementation of a neural network.

2.4.2 Activation Function

When deciding to implement a multi-layer perceptron (neural network) it is imporant to consider and analzye what activation function will work best for your data solution. The types of activation functions have very important influences on the networks' learning speeds, classification correct rates and non-linear mapping precision. Activation functions determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function. Backpropagation is an algorithm which traces back from the output of the model, through the different neurons which were involved in generating that output, back to the original weight applied to each neuron. Backpropagation suggests an optimal weight for each neuron, which results in the most accurate prediction [5].

2.4.3 Decision

In order to decide what activation function fit our model best we experimented with a few activation functions. Firstly we experimented with a logistics function which performed poorly averaging an F1 score around .5. Next we tried using the RELU function which performed significantly better averaging a F1 score around .9. However the best was the Hyperbolic Tangent (Tanh), which averaged f1 score around .95. The advantages tanh provides is that its zero centered meaning it makes it easier to model strongly positive, negative, or neutral inputs. As previosuly mentioned our input (features) are very neutral making this activation function perfect for our model [5].

2.5 REST Service Implementations

REST is an abstraction of the basic HTTP methods (like GET, POST, etc) which is used to build APIs that behave in predictable ways. The basic behaviors of REST services are called CRUD: Create, Read, Update, and Delete [6]. These map to the HTTP methods POST, GET, PUT, and DELETE, respectively. When a server is launched that implements REST conventions, the fields of the URL submitted to the website will contain endpoints and/or parameters which are used to specify the desired behavior.

The interaction between client and server for our service in particular involves the use of 3 yaml specified endpoints: /, /app/run, and /app/retrain. These can be found in our master.yaml file. The "/" endpoint is a simple read action from the client, and a rendered html page with information about our service is returned to the client.

The /app/run endpoint specifies another read action, and returns a rendered html template to the client which will allow a user to upload one or more .csv files containing data which they want classified. When the user clicks the upload button on this page, a create action is used to send the data files from client to server. The files which are uploaded are then run through the model to have their S-cones counted, and then an html file containing feedback is dynamically generated. A rendered version is returned to the client.

Similarly, the /app/retrain endpoint specifies a read action and returns a rendered file-upload html template to the client. The client user will upload files with which to train a new model, and then by clicking the upload button they again generate a create action. These uploaded files are then used to train a new model, which is created, saved, and tested. The metrics calculated as a result of this testing, accuracy, precision, recall, and F1 score, are dynamically written to an html file, which is then rendered and returned to the client.

2.6 Specification

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "cone classifier"
  description: "classify cones from csv data"
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Ryan Danehy and Brandon Fischer"
  license:
    name: "Apache"
host: "localhost:4555"

consumes:
  - "text/html"
produces:
  - "text/html"

basePath: "/app"

paths:
  /run:
    get:
      operationId: scripts.uploadrun.display
      description: "Displays upload file page"
      responses:
        "200":
          description: "Upload page displayed successfully"

  /upload_file:
    post:
      operationId: scripts.uploadrun.upload
      description: "Generate post request to actually upload file"
      responses:
        "201":
          description: "File upload successful"
```

```
/retrain:
  get:
    operationId: scripts.uploadrerun.display
    description: "Displays upload new training file page"
    responses:
      "200":
        description: "Upload page displayed successfully"

/upload_file_retrain:
  post:
    operationId: scripts.uploadrerun.upload
    description: "Generate post request to actually upload file"
    responses:
      "201":
        description: "File upload successful"
```

3 Rookie Fantasy Football Point Prediction

Andrew Gotts, Ethan Japundza, Brian Schwantes
adgotts@iu.edu, ejapundz@iu.edu, bschwant@iu.edu
Indiana University
hid: sp19-222-94, sp19-222-90, sp19-222-92
github: ☒
code: ☒

Keywords: KNN, Fantasy Football, REST, Docker, Yaml

3.1 Abstract

As the number of fantasy football players increases dramatically every year, we saw an opportunity to create a service that will help users draft better teams. While NFL veterans have gameplay for fantasy enthusiasts to evaluate, incoming rookies with a lack of professional experience make it difficult for fans to evaluate whether or not their teams picks will be successful in the NFL. To solve this problem, we implemented a REST service that gets an aggregate of combine and fantasy football data from Google-Drive. Our service then utilizes the K-Nearest Neighbor machine learning algorithm on our data sets, outputting our projections for which 2019 NFL Rookies will score the most fantasy football points based on their metrics from the combine.

3.2 Introduction

Who should be the top pick in this years fantasy football draft? This is a question that has plagued fantasy football enthusiasts since its creation in 1962. But why does it even matter, isn't fantasy football just a game? According to Joris Drayer, a professor at Temple University in Sports Marketing and Analytics, almost 30 million Americans and Canadians actively participate in fantasy sports leagues every year. Drayer goes on to discuss the economic impact that fantasy football has on the sports industry, estimating it to be nearly $4.5 billion [7]. With such a large amount of people and money involved, the technologies created to help players be successful are in a position to revolutionize the fantasy football market. Utilizing the power of machine learning, we were able to create the "Rookie Fantasy Football Point Predictor". A service for those trying to make an educated pick on rookie players with no professional experience.

3.3 Data Set

Our project uses information from three data sets consisting of the 2000-2018_NFL_ Combine statistics, the 2019_NFL_Combine statistics for this year's rookies, and the 2001-2018_Fantasy_Football data for every active NFL player. These datasets include six combine drills:

- 40-Yard Dash: Measures a player's explosion, burst, and acceleration.
- Three-Cone Drill: Highlights a players ability to shift directions at a high speed.
- Shuttle Run: Demonstrates short area lateral quickness, and the speed at which a player can change directions.
- Vertical Jump: Tests for lower body extension and power.
- Broad Jump: Evaluates an athlete's lower-body explosion, and lower-body strength.
- Bench Press: Demonstrates a players strength and endurance.

From these drills measurables are obtained that can serve as a predictor for the rookies effectiveness in the NFL. For certain positions, like wide receiver, the shuttle run is significant as this drill will often determine whether or not the defender is able to get separation between his defender at the line of scrimmage. While for other positions like quarterback, the shuttle run has less importance as they are rarely making quick and decisive cuts while playing in NFL games [8]. Due to the difference in how each drill can serve as a predictor for different positions, we found the best way to analyze the incoming rookies within the 2019_NFL_Combine dataset was to compare their performance to previous NFL players within the 2000-2018_NFL_Combine dataset. From this comparison, our goal was to both rank and predict the average fantasy points-per-game for the incoming rookie class based on how their comparisons performed in the 2001-2018_Fantasy_Football dataset. We found the best way to make this correlation was to employ the K-Nearest Neighbor machine learning algorithm for our service.

3.4 KNN Algorithm

The machine learning technique KNN (K-Nearest Neighbors) is a technique often used for classification or linear regression predictive problems. The algorithm relies on feature similarity, or the process of checking how an input sample will resemble the training set [9]. Using the Euclidean distance formula, KNN calculates which features of the input sample are nearest to the training set. Figure 4 is a graphical illustration of this process.

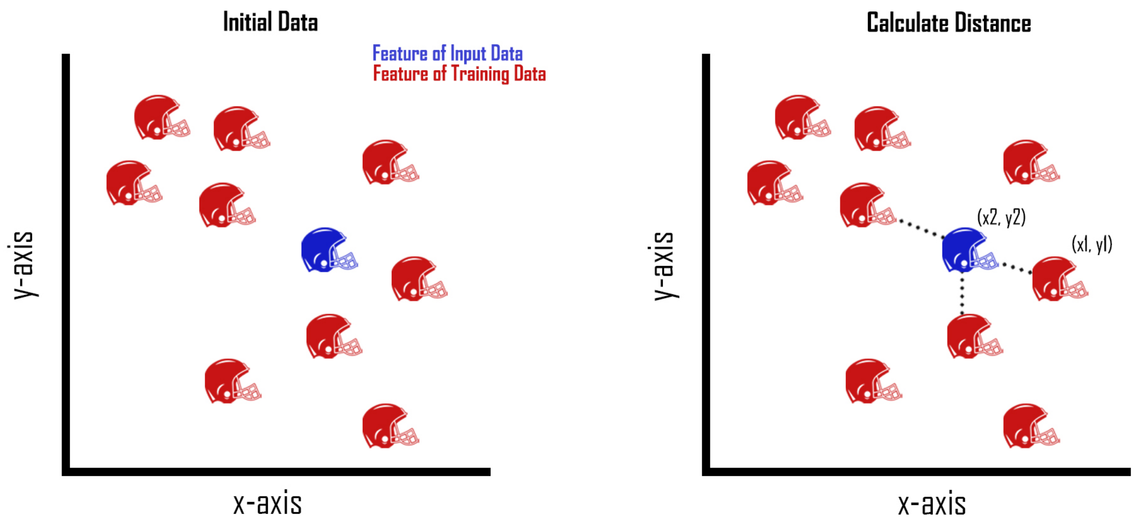


Figure 4: Euclidean distance formula

The Euclidean distance formula, Equation 1, is used within KNN to calculate the distance between a feature of the input data, blue helmet, and k instances of that same feature within the training data, red helmets. Choosing the value of k is dependent on the training datasets used and the desired outcome. Compared to other machine learning algorithms, KNN has an advantage as it does not make assumptions about the data it uses, it is simple, highly effective, and it is versatile. However to its disadvantage, the KNN algorithm requires that we store all of the training dataset, which will often cause the algorithm to be computationally expensive and slow [9].

$$EuclideanDistance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

Before our implementation of the KNN algorithm, we found that choosing a value of three to represent our k would best optimize the results of the program. In the python file k-nearest, we call and train the KNN algorithm with the NFL combine results from the last 18 years. Using the incoming rookies 2019_NFL_Combine statistics as the input data, the program calculates the Euclidean distance between each feature of the input data and the accumulated combine results of previous years. This calculation yields the three closest veteran players that each NFL rookie performed the most similar to during the combine. While the results of our implementation are accurate and informative, the speed disadvantage of the KNN algorithm becomes apparent within our program. Despite making adjustments to speed up the algorithm, such as only running KNN on the instances of the input data called for opposed to running it on all of the input data and filtering it afterward to get a tailored output, the program still runs slowly. This is due to the KNN algorithm requiring us to store the entire dataset before it runs computations on it. Despite our best efforts of optimization, our program still has a run speed of about five seconds.

Another disadvantage of the KNN algorithm and other machine learning techniques is their tendency to un-intentionally weight certain features of datasets due to size and scale differences. To account for this, we have to normalize the datasets we pass into the algorithm, creating an equal "weight" for the Euclidean distance to calculate. In Python using the PANDAS library, or Python Data Analysis Library, allows us to do just that, and easily import the csv's needed to execute the KNN algorithm.

From this library we use the read.csv function to read the NFL csv, comma-separated value, files into a data frame. A data frame is a two-dimensional tabular data structure with labeled axes, in which arithmetic operations align on both row and column labels. We use this data frame to isolate only our numeric features within the dataset, allowing us to quickly use only the features from the NFL combine that are measurables. As discussed earlier, in order for our output to be accurate the numeric features within each dataset must be normalized to create an equal weight for each combine measurement. Using the PANDAS data frame we achieved this by taking the numeric features and subtracting them from their mean. Once this was done we divide them by the standard deviation of the dataset, resulting in a very neat and evenly weighted data frame to pass into the KNN module.

3.5 Implementation

The core of this project is written in Python, which allows us to package all components needed to run our service easily. The use of "in_it" files creates a sensible directory structure that ensures a user can import both functions and data if they see them useful for personal applications. The versatility of python enabled us to create functions that execute the machine learning on our NFL datasets, along with a server that hosts these functions tied together through a REST service.

REST, or Representational State Transfer, is a style of architecture used when creating web services. From the textbook, REST is described as being "based on stateless, client-server, cacheable communications protocol." Applications using this architecture typically use HTTP protocol with basic methods like GET, PUT, and POST. These methods allow applications using the REST architecture to perform the four CRUD operations, which are to create, read, update, and delete resources. User created functions can be combined with these operations to create a cloud service that completes predefined tasks between a server and a client. However, to establish this direct link between a server and a client a YAML file is required [10].

A YAML file is written in a readable data serialization language that can be used to configure endpoints for a server. These endpoints are used to dictate what URLs are valid within the server, and what data, or data types, will be displayed for a particular path. YAML also allows specifications to be added to endpoints such as operations, HTML responses, and HTTP methods. Future additions to endpoints are simple to add without affecting current endpoints, which makes future updates straight forward with few worries about breaking past versions. This provides flexibility that is complemented by an easy to read syntax.

Our project uses REST architecture, in conjunction with a YAML file, to package our python modules so that they are able to run on a local machine. Within the YAML file, there are predefined paths a user can enter for different purposes. For our project, the path can be altered by changing "/results/" to view fantasy football point predictions based on a given position.

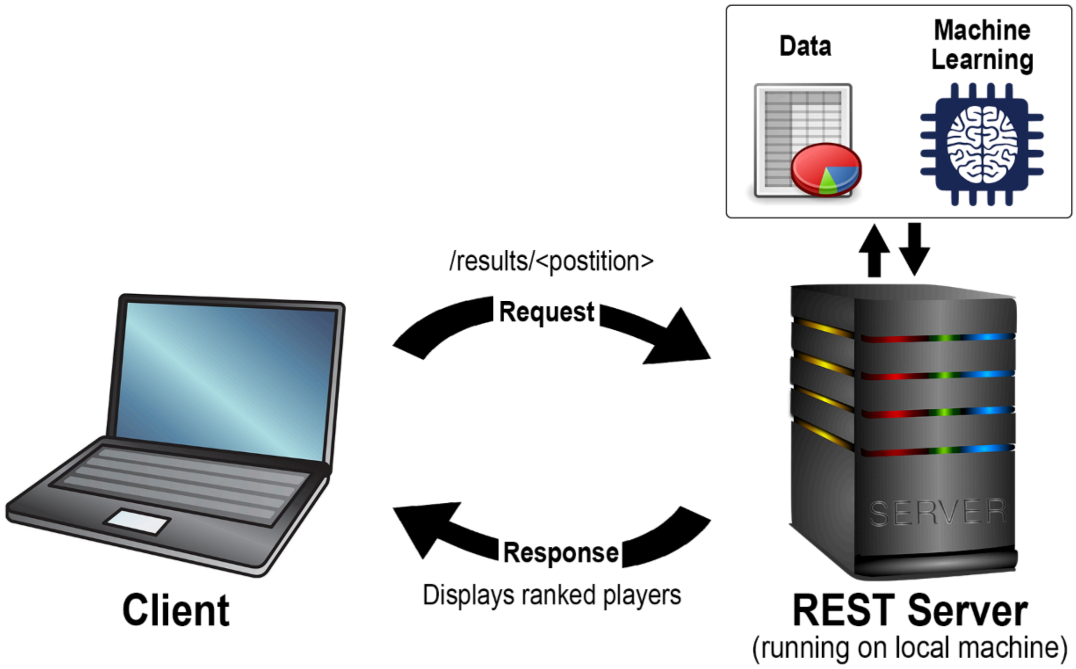


Figure 5: REST Service

Visualized in Figure 5, when an offensive position is entered, the HTTP request is then sent to the REST server. From here, our service calls the python function ff_prediction(position) within the nfl-analysis.py file. First datasets are retrieved from a Google drive. Next, the machine learning is executed and the players are ranked through our comparative analysis and exported to a csv file. We then convert the final csv file to a html table, where it is printed to the users screen at the endpoint defined in the yaml file. While this is functional, we aren't able to ensure consistency across multiple development and release cycles, which highlights the advantage of using a software like Docker to standardize the environment of our project.

Docker is a tool that allows users to create, deploy and run applications within virtual containers. This technology is extremely useful when developing applications because it allows one to combine all of the needed components into one package [11]. Docker also ensures parity, which allows your images to run the same regardless of the server or laptop it is running on. This eliminates bugs that will occur on only certain computers, creating a standardized environment. We utilized Docker within our project by implementing compatibility through the use of a Dockerfile and a Makefile which makes it easy for users to build the Docker container, run the server within the Docker container, and remove the Docker images. The inclusion of the Makefile simplifies the process for users looking to run our service and remove it once they are finished with it, ensuring no compatibility issues. The user only needs the Dockerfile and the Makefile to use our service to its fullest extent.

3.6 Limitations

As mentioned before, the NFL combine has six tests that measure speed, agility, and strength. We used that data in conjunction with each players height and weight when finding their “nearest neighbors.” However, not all players, rookies and current, completed every drill at the combine. This can result in inaccurate rankings, as KNN is only finding comparisons within one drill. Unfortunately, this gave players who skipped drills a boost in our rankings, as they tended to dominate in the the drills they did choose to participate in. Another hindrance is that we had no way of knowing what NFL team each rookie will be drafted to, and whether or not they will be becoming a starter or second string player. The role the rookie has within his team is a huge factor in Fantasy Football, and not being able to take this into account creates a limitation on how accurate our model can be. Going forward, one way we could attempt to increase the accuracy of our predictions is to compensate for both of these factors, and to utilize rookies college football statistics. While using college statistics was in our initial plans, we ultimately decided that the differences in conference and division could provide skewed data resulting in inaccurate predictions. This would not be impossible to account for, but it was not realistic to achieve this in the timeframe we were given.

3.7 Conclusion

As the number of Fantasy Football players continues to increase, we set out to answer the age old question, who should be my top pick in this years fantasy football draft? To help our fellow fantasy football players answer this question, we provide users of our service with a way to make educated draft picks on rookies that are likely to succeed in the NFL. To accomplish this goal, we used nearly 20 years worth of NFL combine and fantasy football data to create a service using REST, Python, Yaml, and Docker. Utilizing these technologies along with artificial intelligence, we were able to successfully rank rookies by their position, accompanied by our prediction of average fantasy points per game.

We hope to test the accuracy of our service throughout the course of 2019 to see if this years rookies will hold up to our predictions. If we were to address some of our limitations, adding factors such as the role a rookie will play on his team, we would be able to continuously update our service and make adjustments. Analyzing these results would help us make it more accurate for next year's draft, and if we find our algorithm to be successful we have an avenue for deploying it into the real world. Providing users with educated sports picks through machine learning could revolutionize fantasy sports; and in a market that has nearly a $4.5 billion impact across the sports industry, this information has the opportunity to produce serious revenue. We hope to continue building on top of this foundation, and refine our model with hopes of eventually tapping into this market.

3.8 Specification

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "NFL FF Points"
  description: "A service that provides Fantasy Football point predictions for u
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "NFL FF Prediction Service"
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/cloudmesh/nfl-analysis/v1"
paths:
  /results/{position}:
    get:
      tags:
        - NFL
      parameters:
        - name: position
          in: path
          description: position
          type: string
          required: true
      operationId: nfl-analysis.ff_prediction
      produces:
        - text/html
      responses:
        "200":
          description: "FF data"
          schema:
            $ref: "#/definitions/NFL"
definitions:
  NFL:
    type: "string"
    properties:
      model:
        type: "string"
```

4 Analysis of soccer data with kmeans

Jesus Badillo, Xandria McDowell, Ben Yeagley
jebadi@iu.edu, xmcdowel@iu.edu, byeagley@indiana.edu
Indiana University Bloomingtion,
hid: sp19-222-92
github: ⊠
code: ⊠

4.1 Abstract

Data collected by wearbale devices used by the Indiana University Women's soccer team was used to classify the six unique offesive and defensive positions excluding the goaltender. Successful classification of positions from wearable data using machine learning techinques could provide talent evaluators and coaches with an extra tool to help increase on field productivity. The k-means algorithim was used in order to classifiy the data from wearble devices into six unique positions. This is the first step towards creating a workflow for using machine learning as a talent evalution tool for soccer players. Once the ability to successfully classify data from wearable devices into unique positions additional layers of complexity can be added, like comparing professional athletes or collegiate athletes to prospects. A framework for the implemtnation of classfiying soccer postions has been implemented with Scikit-learn libraries in python3. OpenAPI Specification was used in order to create a service that exploits the Scikit-learn libraries through an easy user interface.

4.2 Introduction

Can data gathered from athletic practices and games be used to create benchmarks for how an athlete playing a specific position should perform? Recent advancements in wearable technologies has allowed data from sensors to instantly improve how machines and people perform. These advancements are now being used in sports to make computations to player data to get future insight on how to improve the overall team. For example, the popular movie Money Ball focused on how analytics can be used to assemble a competitve professional baseball team. This type of analysis can show users what datapoints are shared by top players allowing coaches to pick their players accordingly. Soccer uses sports data to improve the physical attributes, such as stamina and pace, of each individual player since it is a more technical sport that cannot just use quantitative data to make decisions. The field of sports analytics has been changed by machine learning techniques that have been adapted to use the hard numerical data received from edge-computing devices to improve teams.

4.2.1 Sport Analytics

The rapid innovation of technology has infiltrated many fields including sports analytics because it has created more data that can show coaches what athletes need to improve on. Edge computing is the technology that has led to the massive increase in more refined sports data because of its ability to provide a real-time numerical representation of what happened on the field. With this numerical data, actions can be taken to rapidly improve each player specifically on the areas where they need immediate attention. Edge computing is computing that is done near the source of the data, and it uses the cloud's ability to compute from a remote source to get instant data from a device. Edge-computing has allowed sensors to get more accurate data without disturbing the players which has given many sports a new way to get an advantage over their competitors.

4.2.2 Sensors in Sports

Sensors have become more prominent in sports due to recent advancements in biomedical engineering which have made sensory technology better to create more informative data. These new innovations in sensory technology can now measure data points such as "temperature, oxygen saturation levels, and heart rate (SpO2) through photo optic sensors in wearable rings and wrist devices"[12]. These new advancements are necessary because now, more than ever, "the differences between athletes are becoming more and more slight"[12] which has led coaches and trainers to look to data analytics to get an advantage. By putting small sensors into players' training equipment, we can detect important aspects of playing a sport such as muscle exertion, heart rate, and respiration. Sensors can provide immediate feedback to the athletes. For example, when attached to player attire, "accelerometers and conductive materials can measure posture and provide real-time feedback to athletes so they can perfect their form"[12].

Soccer has recently began to adopt the use of technology in many different ways such as the Adidas MiCoach ball, which is a regular sized soccer ball with a built in sensor that can detect the power, spin, and accuracy of a players kick. This ball allows players and coaches the to see how the athlete shoots the ball through an iOS or Android device to improve the players shooting form. This type of sensor is mainly used for beginners, professionals tend to focus on the more physical side of data collection such as wearable sensors that track physical stats. The Catapult Sports Playertek is a compression vest with a built-in sensor that has an accelerometer, magnometer, and a GPS that can track athletes' performance during practices. The IU Women's soccer team uses sensors like the Playertek to get athlete data because it allows the coaches to know what physical attributes to track for each player when they are training or playing.

4.3 Soccer Dataset

Practice and game data from the IU Women's Soccer team was used in this analysis. The results of this analysis aims to leverage "sensed" athlete data to show coaches and players if players are performing optimally. The Women's team data has 34 individual columns such as sprints, accelerations in varying ranges, and heartrate that are weighted based on how much time the individual played. Some data points, such as sprints, can be used to improve the athlete in a certain aspect of their sport, in this case by increasing the number of sprints they have during a game. Using data in this way can help the player improve their speed, but it will do very little to improve their overall ability. Instead the Kmeans Clustering method will show the entire team how they compare to each other and the ideal player of their respected position, so that the entire team can improve. By using Kmeans on the data each player can improve how they play, not just how fast they can run. The Kmeans algorithm is optimal for this dataset because it uses eight different datapoints to determine how well each player is playing their position compared to how they should be. This method provides a literal target from which players can see which areas they specifically need improvement on to reach the centroid, or the ideal player for that position.

Clustered data can be used to compare players to other players who play the same position in different divisions, levels, etc. This method can sometimes be inefficient because what is *good* can vary a lot in soccer, due to it being based on skill rather than physical attributes. For example, some defenders do not run very much at all, but instead rely on their ability to read players movements to stop the opposing players from getting past them. This would be a great data point that could likely be used to create more accurate models that can help the players distinguish themselves from the other positions in terms of how they should be performing. The problem is that sensors are limited to numerical data, so this type of ability cannot be measured. To avoid any of these limitations, the Women's Soccer team data has many of the same data points measured in different ranges. For example, the *Number of Accelerations* category has eight different ranges which are either a deceleration or an acceleration. This way of splitting data could transcend the inability to measure the *intangibles* because it measures the same data points at different instances which tend to vary by position.

The Women's team data was obtained from the 2018 season where they faced off with many other Big 10 schools. There were eleven datasets which each contained a row for each of the players where you could see their names and all of their data points for that particular game. Since each dataset had only 23 data points we decided to join all of the datasets into one big dataset where the players' names were removed and replaced by the player's positon into a column called *Class*. The original datasets that were given to us did not contain how many minutes each player played, which could alter how the kmeans algorithm clusters the players. The algorithm would likely cluster the players wrongly if the *Minutes Played* column was not added because it would give the players who are substitutes higher values without taking into accountthe amount of time that they played during a game. In soccer, the amount of time you play can especially effect many of your stats because the sport is largely based on endurance with a few short bursts of sprints. The substite players who had no game time were removed from the dataset because they could alter how the algorithm clusters a good stat vs. a bad stat. Once the extraneous data points were removed there were only 170 rows left which all had a label of either Mid-fielder, Forward, or Defender. The label was used as the prediction variable to give insight as to where the kmeans-computed clusters should be centered. The Kmeans algorithm uses the labels to predict the outcome of each of the 170 rows' stats against the kmeans-computed stats.

4.4 Algorithm Discussion

4.4.1 K-means

K-Means is an unsupervised machine learning algorithm that takes an unlabeled dataset and finds groups within data without defined categories. Unsupervised is learning from data that are unlabeled meaning you don't know what the values of the output data might be. The idea is to look at the average or mean values that can be clustered and how they can be related to the k groups. K is also known as centroids which is a data point at the center of each cluster. Each cluster is placed in a certain position because depending on how close the clusters are to each other will change the results. Which is why the placement of the cluster are important because the layout of centroids will determine the results. In K-Means the centroids are randomly selected and then used as the cluster of each point. After the centroid is placed then all the data point is split up into the correct centroids based on the data point that they are. Now that the data points have a home the next step is re-calculation of k to optimize the position of the centroid. This process of optimization continues to happen until centroids have stabilized or a certain number of iterations have occurred. When using this algorithm, the hope is that the results can explain why common parameter values are in the same group. This algorithm takes numerical values or data points that describe a coordinated value within a data set and cluster them according to the given k. For example, K- Means is beneficial for this project because it's able to focus on a particular data set for the women's soccer team at Indiana University (IU) and compare that to users data. K-Means coordinates can represent and describe many different things such as elections, medical test, sports teams, or wine data. Typical you use K-means on a dataset to get the model of the data, this is used to see the behaviors and analyze the patterns the data set. When using clustering methods variance becomes an important topic trying to understand the math behind this algorithm. Variance is a measurement of how far data values are from the mean. By look at the variance equation, it shows that it is a measure of the square difference between the data points and the mean value and then the average. When using the K-means algorithm one of the key points is to measure the closeness of the data to its average which led into cluster variance which divides the data into clusters, each will have its own average. While there are many different machine learning algorithms K-Means worked best because it takes a unsupervised data set which was need in this case. While K-Means gives an optimal cluster model, spectral cluster is better. The K-means objective function is given in Equation 2.

$$\sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^{(j)}-c_j\right\|^2 \tag{2}$$

4.4.2 Spectral Clustering

Before landing on K-means to be the machine learning algorithm, spectral clustering was the set machine learning algorithm for finding out how close people were playing against Indiana University womens soccer team. A commonly used algorithm for classification that has become popular in recent years is spectral clustering [13]. This algorithm can be effectively solved by using standard linear algebra methods and, software. For the most part, it is known to outperform K-means which is what was seen with this data set. Spectral clustering uses the data points as nodes of a graph, which are then treated as partitioning points and mapped to a low-dimensional space making it easier to form clusters. Spectral clustering follows an approach where if the points are extremely close to each other they are put into the same cluster, even if the distance between them is within a two point range if they are not connected they are not clustered. While k-means points can be within the same range and still fall within the same group because it is measured by the distance between the data. This data set has many points where they are considered within range of each other which is why spectral clustering gave better results. In spectral clustering the data is connected within a way that leds to the data points falling within the same cluster. The problem arises when using spectral clustering because the project aims to show people how they played compared to Indiana University soccer players but when trying to show their data and the soccer players data on the same plot issues arise. This happens because you can not get the labels using spectral clusters so the use would not know where they where. spectral equation below.

$\left(\frac{1}{2}\right)\sum_{i,j=1}^{n} w_i j(f_i - f_j)^2$

4.4.3 Dimensional Reduction

Dimensional reduction was something that was needed for this data set because there were many points within one players statistics. Having multiple points, was too much for K-means to give explainable results. Dimension reduction allowed the data set to be reduced to a certain number of random variables by obtaining a set of key variables. Looking at dimension reduction from the math perspective there are the non-linear and linear methods. Typically, the Linear method is used because it is easier to implement. In the linear calculation results in each variable being apart of a linear combination of the initial variables such that $k \geq p$ see Equation 3.

$$S_i = w_ix_i + ...w_ipxpforI = 1, ..., k \tag{3}$$

where

$S = Wx$

Wx represents the weighted matrix in linear conversion. $Ap * k$ is the same as $x = As$ which are new variables or s identified as hidden or latent variables according to [14]. When using the matrix X in terms of nxp Equation 4 is applied.

$$S_ij = w_1jW_1j + ...w1pXpJforI = 1..., kandj = 1..., n \tag{4}$$

According to 14 this is the easiest linear technique to use.

4.5 Results

Normally a clustering model cannot be scored for its accuracy; there are no true labels in unsupervised learning to compare with the clustering models predicted labels [15]. However, in our case each of the fitness data points had the player name associated with it, so we were able to replace each name in the dataset with their position on the field (the predicted value). Converting these positions to numeric representations gave us a list of true labels to use for scoring. We found each players position by taking the position listed on the team roster page. Of course, some players may play in different positions throughout the season, so some of the labels may be incorrect. Our labelling process was the best we could do with the information available. We focused on three evaluations for our model: completeness, homogeneity, and v-measure. Completeness is a measure of how well each class was clustered together [16], while homogeneity scores the clusters based on what degree in contains only one class [17]. The harmonic mean of the completeness and homogeneity scores gives the v-measure score [18] [19]. Each is considered perfect if their score is 1.0. Conversely, 0.0 is the worst possible score. The results of our clustering model can be seen in Table 2. had a completeness score of .260, homogeneity score of .227, and a v-measure score of .242. This means our model was more effective at clustering the classes together than having each cluster contain purely one class. Overall, the model scored poorly. Its scatter plot [20]:

Results of clustering: ⊠ not following image standrads

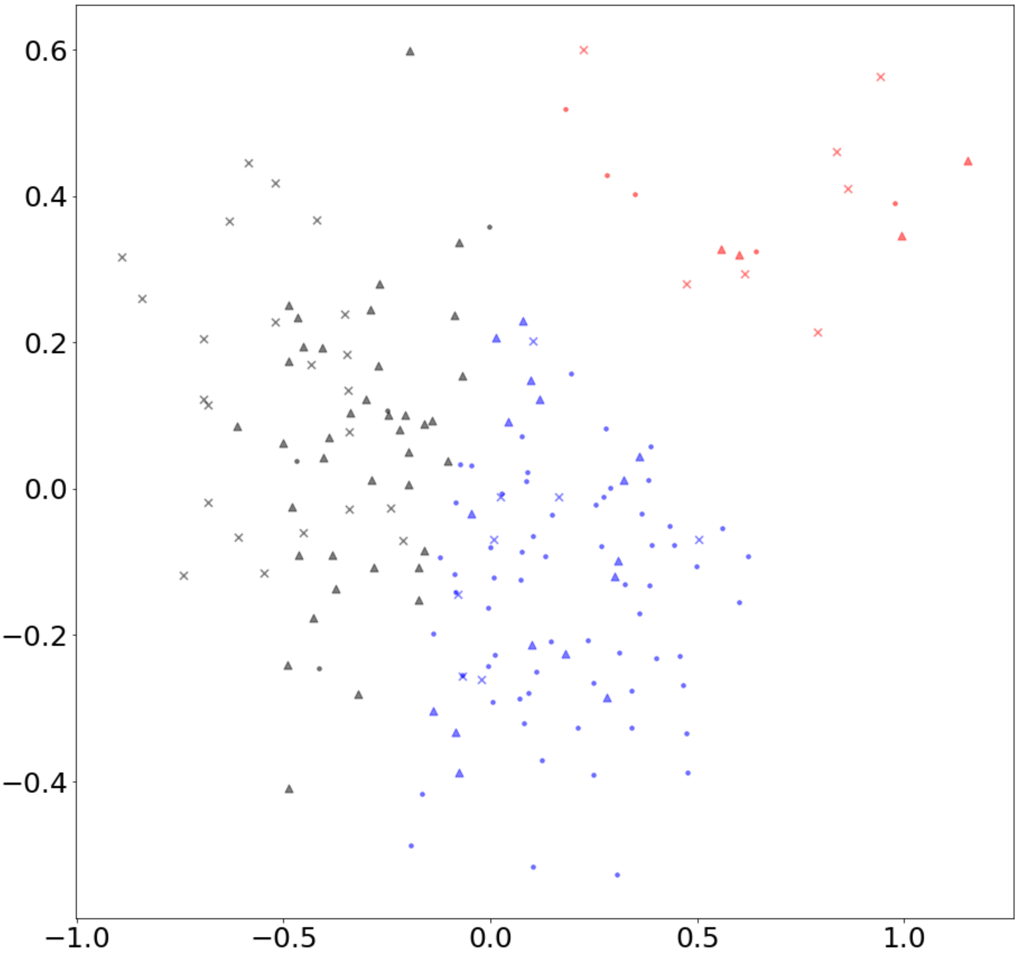


Figure 6: Clustering scatter plot

Looking at the scatter plot showing the clusters and the data points true labels, we can see that the poor score can be mainly attributed to the general failure to classify forwards (x's) correctly. The defenders (dots) and midfielders (triangles) have fairly defined clusters themselves, and viewing the scatter plot without the any forward data points makes it even more apparent.

Clustering results, forwards not displayed: ⊠ not following image standrads

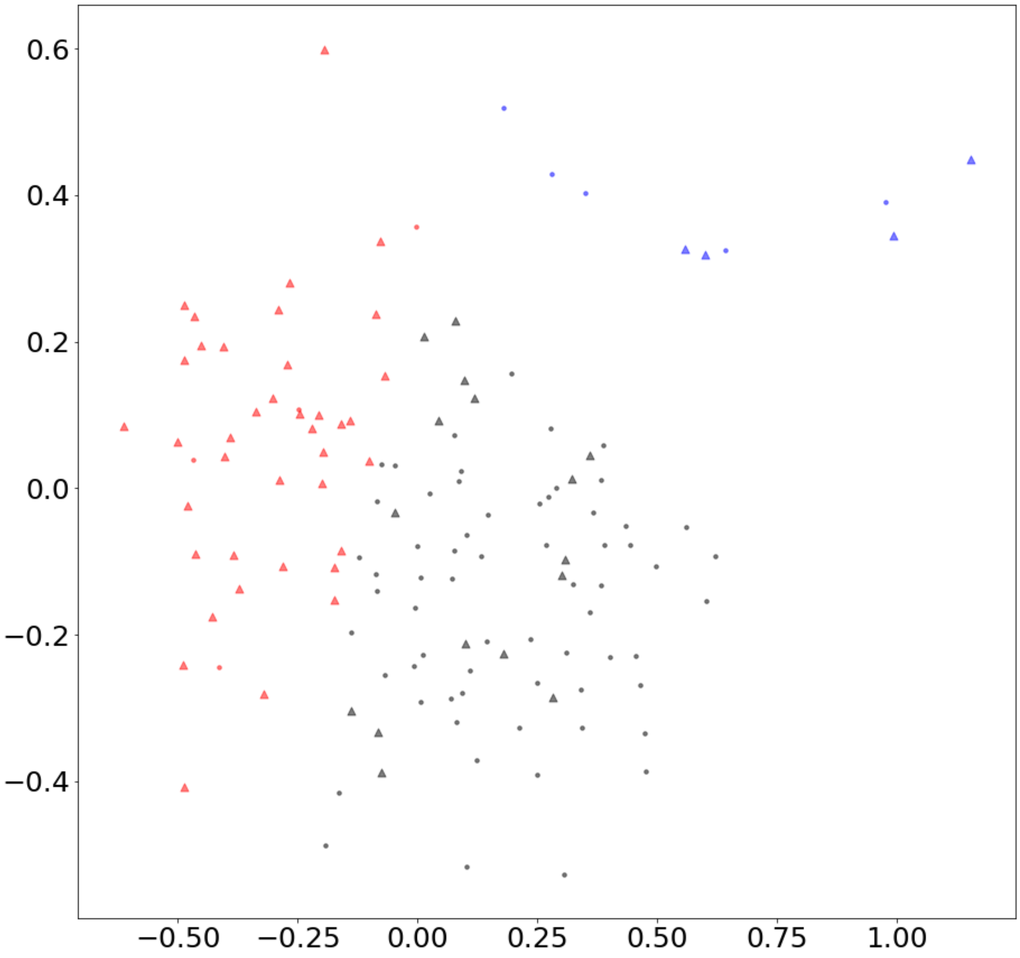


Figure 7: Clustering scatter plot

The black cluster was split almost 50/50 between forwards and midfielders despite them being decently separated groups, mainly because the model identified the outliers of the dataset as the red cluster. It's possible that if the dataset was cleaned up by removing the outliers before clustering, the model would've separated the classes much more accurately. Outliers as defined by our boxplot are shown here [21] [22]:

A Boxplot of the data after normlization is given in Figure 8.

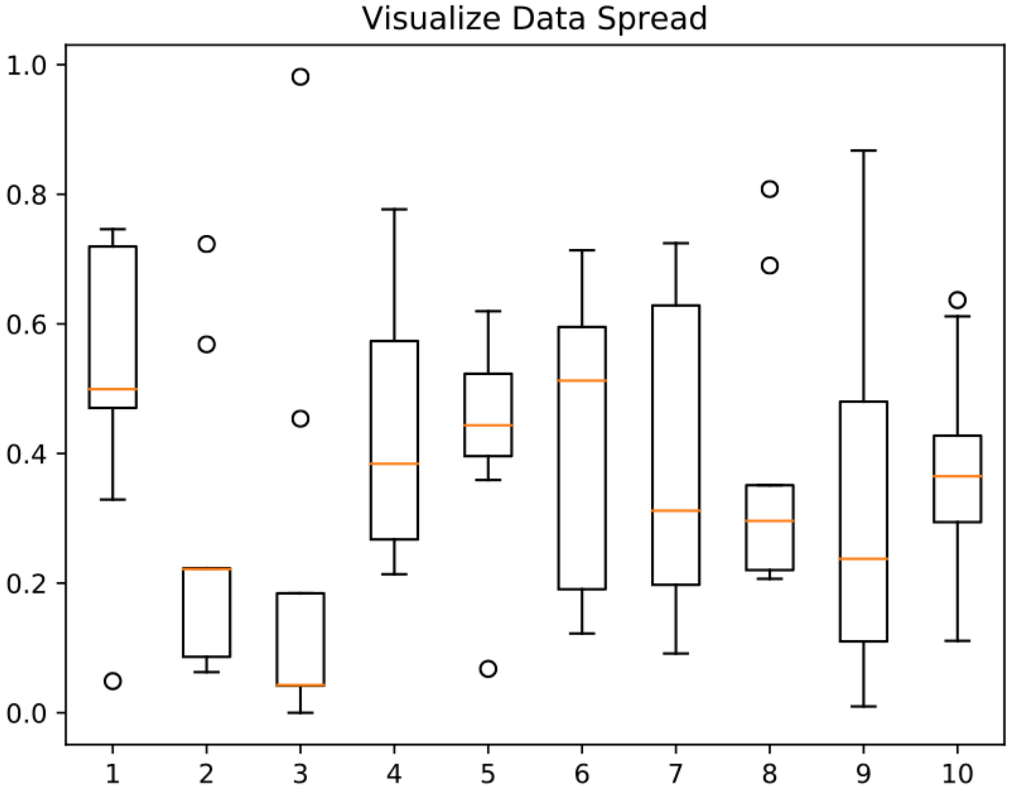


Figure 8: Boxplot after normalization

Our method for removing the outliers could have involved calculating a Z-Score for each data point, which gives the amount of standard deviations a point is from the mean of the dataset, and then getting rid of points with a Z-Score above a certain amount of standard deviations [23]. However, it's important to determine if the points in that top-right cluster would even qualify as true outliers or if they are just points the clustering model is unable to predict. We might need to investigate those points in the original dataset to see if there is a discernable pattern to them because theoretically, the data shouldn't have any anomalies unless the measuring device malfunctioned. It could even be a specific game that led to those points being separated; the game could have been much more intense than normal, leading to higher numbers for each player that participated. Disregarding them just to improve the models accuracy may not be the right choice.

Table 2: Results of our k-means analysis

completeness	homogenitiy	v-score
0.260	0.227	0.242

4.6 Analysis

A necessary question to ask is: is the low v-measure score of the model due to possible errors in the dataset like those discussed above, an incorrect choice of machine learning method, a general unpredictability of player position with the type of data given, or some other reason? To examine the second possibility (wrong method), we split up the data into training and testing using train_test_split from sklearn, and then we added a variety of other classification techniques to the project code [24]. The first technique was Multinomial Logistic Regression, a method that uses a linear combination of the predictor variables to model the dependent variable (position) [25]. The model had an accuracy score of .63 for our training data and .60 for our test data. The next technique we employed was a Decision Tree Classifier, which handles multi-class predictions naturally. Decision trees use the observed variables from the dataset to make choices (branches of the tree), leading to a conclusion about the predicted value (leaves of the tree) [26]. We gave our decision tree a max depth of eight as its only parameter. It had an accuracy score of .80 for the training data and .65 for the test data. Next, our K-Nearest Neighbors Classification model functions by taking a vote of the points around the point to be classified and assigning the point whichever class that had the most votes [27]. The number of neighbors used can be specified in the function call, and we found eight neighbors worked best. Our k-NN model had an accuracy score of .76 for the training data and .74 for the test data. We have a Linear Discriminant Analysis classifier as well, with the number of components set at two [28] [29]. Its accuracy score was .69 for the training data and .60 for the test data. We also had a Gaussian Naïve Bayes classifier despite the fact that our data doesn't have features that can be considered independent, but it still serves the purpose of comparison to the clustering model [30] [31] [32]. Its accuracy score was .73 for the training data and .63 for the test data. Finally, we had an SVM classifier added to our project code, another method better suited for just two classes but used anyway [33] [34]. Our SVM model had an accuracy score of .58 for the training data and .63 for the test data. Overall, the highest scoring model for the test data was K-Nearest Neighbors, yet its .74 score for the test data still was not very high. It is difficult to make a direct comparison between the k-NN model and our original clustering model for multiple reasons. Their scores don't really represent the same idea, as accuracy isn't well defined when it comes to clustering. Clustering is more generic than k-NN classification too, because its goal isn't to predict classes for each individual data point. Instead, the goal of clustering is to group the data into distinct sets and see how these sets align with real-world observation and truth. Failed clustering (or poor clustering in our case) means our dataset couldn't fully differentiate positions with the variables measured. An aspect sorely needed to further separate the classes would be GPS data to find the average location on the field for each player, or a heat map of their movement throughout the game.

Table 3: Machine Learning Table of Reults

Machine Learning Method	Accuracy (training)	Accuracy (testing)
Logistic Regression	.63	.60
Decision Tree	.80	.65
K-Nearest Neighbors	.76	.74
Linear Discriminant Analysis	.69	.60
Gaussian Naive Bayes	.73	.63
SVM	.58	.63

4.7 Specification

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "Clustering Athlete Data"
  description: "A simple service that gets game data from a cloud storage servic
  termsOfService: "http://swagger.io/terms/"
  contact:
    name: "Cloudmesh REST Service with AI"
  license:
    name: "Xandria McDowell, Ben Yeagley, and Jesus Badillo"

host: "localhost:8080"
basePath: "/project19"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:

  /data/output/<output>:
    get:
      tags:
        - DATA
      operationId: functions.download
      description: "Downloads the dataset from the URL"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Data info"
          schema: {}

  /data/kmeans/<datafile>:
    get:
      tags:
        - KMEANS_PLOT
      operationId: functions.kmeans_plot
      description: "Filter the dataset, normalize, and perform Kmeans"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Data info"
          schema: {}

  /data/user/<datafile>:
    get:
      tags:
        - USER_DATA
      operationId: functions.user_plot
      description: "user data"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Data info"
          schema: {}

  /data/boxplot/<filename>:
    get:
      tags:
        - USER_DATA
      operationId: functions.boxplot
      description: "user data"
      produces:
        - "application/png"
      responses:
        "200":
          description: "fig info"
          schema: {}
```

5 Tetris Score Analysis Server

Zach Levy
zwlevy@iu.edu
Indiana University
hid: sp19-222-97
github:☒
code:☒

Keywords: Tetris, linear regression, decision tree, gini coefficient, correlation coefficient, determination coefficient, Python

5.1 Abstract

Tetris is a popular video game played by many professional and casual players worldwide. Using data recorded from players in from the Tetris World Championship and analyzing the qualities between various factors such as playtime, score, level, and more, the aim is to possibly extract meaningful relationships and evaluate them using standard regression models. The usage of a linear regression model and decision trees are used to explain if any relationships between these exist, and whether said relationships are meaningful. From the extrapolated data, the aim is to use statistical coefficients, such as the correlation coefficent and Gini coefficient from the models to do this.

5.2 Introduction

Released originally on the Electronika 60 in the USSR on June 6, 1984, Tetris has came from a small programming project from Soviet game designer Alexey Pajitnov to one of the most world-renowned puzzle video games of all time. Although deviations have spawned from the game, the primary rules of Tetris are rather simple. Blocks, arranged into sequences of four called tetrominoes, move vertically from top to bottom. The player can reposition and reorient the blocks, and the goal is to get a set of blocks all on a bottom line, in which the blocks will disappear and add to the score. After many of these occur, the game will speed up, making it harder to arrange blocks correctly. The game is over when the blocks reach to top of the screen. Ported to a wide variety of platforms, Tetris has become a part of popular video game culture and has sold millions of copies. Psychological studies done on people who play the game have lead to the discovery that people who play for prolonged periods of time may see mild remote memories of the images of the tetrominoes moving [35]. Additionally, some studies suggest that Tetris may help reduce mental stress and help individuals cope with post-traumatic stress disorder [36].

5.3 Design

The primary design for this project comes in two stages:

1. The creation of a regression model that most suitable fits the dataset. This should most likely be a linear regression model or a data tree.
2. The creation of a RESTful server that could retrieve the raw data sets, perform analysis, and return the data to a requested user.

The first part of the project required me to program a way to retrieve the transcribed data. Due to the data on the website being only images of spreadsheets rather then the spreadsheets themselves, this was neccessary [37]. By importing the Python libraries csv and requests, the program can retrieve this data from a Google Drive downloadable link as TETRIS_DOWNLOAD.csv.

By using a decision tree and a linear regression model, we can better understand which relationships between variables are meaningful and which are not. The analysis of the data was first done with the usage of a linear regression model. For this, I took the features to be year, rounds won, and composite score. The label in question was the resulting ranking of players. The second model used was a data tree classifier. This would help with the previous linear regression model, since we can observe how the Gini coefficients - numbers that represent the inequality distribution of data tree nodes that help color the importance of certain variables - are resulted from the usage of a decision tree. Gini coefficients are assigned to each branch and leaf of a decision tree and evaluate the decision inequality when evaluating whether an entry is one or another. If the nodes expectedly go to zero, it is a positve indicator that the model fits well.

5.4 Architecture

There are two primary structures within this experiment. The first is the linear regression model module. This module can retrieve data from a Google Drive link and downloads it locally. Using a Representational state transfer (REST) service, it can send over results of analysis on the data set. It does this by running HTTP requests to the server that contains an API to manage the endpoints of communication and send HTTP responses back to a client. The scipy library makes it very simple to perform linear regression. It gives three distinct graphs - divided into subplots on a single image - and uses matplotlib to make graphs that are easily human-readable.

1. Rank vs. Year
2. Rank vs. Score
3. Rank vs. Rounds Won

As for the data tree module it will simply read the data the same way as the linear regression module and create a .dot chart from the results, which can be saved remotely just like the graphs within the linear regression module. The Python library graphviz makes for easy use of creating the decision tree to reflect the data used within this experiment, while libraries such as csv, io, os and requests allow the modules to download and read the extrapolated data with ease.

5.5 Dataset

The data used within this project was primarily extrapolated from data of the winner results from the Tetris World Championships. Due to the fact that previous years represent data as only image screenshots of spreadsheets, the data had to be manually transcribed into other comma-separated value spreadsheets. The data was extrapolated from the spreadsheets into a Google Spreadsheet and saved into a comma-sepearted value file. Extrapolated entries included a player's name, composite score, rank, rounds won, and the year the game took place.

5.6 Results

To put the results into better interpration, there are various mathematical ideas that can help explain the actual importance or usefulness of relationships shown. Most importantly when it comes to our experiment is the following for linear regression models and data trees.

- **Correlation Coefficient.** Denoted by *R*, a correlation coefficient measures the strength of linear relationships of a scatter plot. If *R* is close to -1, then a strong negative linear relationship exists. Likewise a +1 indicates a strong positive linear relationship. However, if *R* is closer to zero, then it indicates that a linear relationship isn't very present [38].

R is calculated as follows: $R = \frac{\sum((x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}$

- **Determination Coefficient.** Denoted by *R^2*, the coefficient of determination how much variation of data has been explained by the model. The closer this value is to one, the better. An *R^2* that is less then zero indicates that it is a poorly fit model, and the reason this is such is that we have used bad constraints or a made poor choice in model 39.

R^2 is calculated as follows: $R^2 = 1 - \frac{SS_{Res}}{SS_{Total}}$

The following table shows various points of data found based on the graphs.

Table 4: Linear Regressor

Graph Type	Slope	Intercept	R Value	R Squared	P Value	Standard Error
Rank vs. Year	0.480	-950.059	0.100	0.010	0.129	0.129
Rank vs. Score	-8.474	24.648	-0.492	0.242	1.495	0.001
Rank vs. Rounds Won	-6.928	29.911	-0.896	0.802	4.607	0.226

With the correlation coefficients of the graph, most of the relationships we found did not have much importance. As we can tell, the *R* value of Rank vs. Year was closer to zero, indicating that there is not any strength in a linear relationship with the two statistics. In the Rank vs. Score graph it indicates a middle-leve of strength with *R* = -0.492. This suggests a negative linear relationship that isn't so strong, but it is expected since ranks are actually lower in number, meaning 1st place is actually the highest, but 32nd place has a higher number. That is why when discussing them we look at the *R^2* value instead, which is positive. This carries over when discussing with Rank vs. Rounds Won, as *R* = -0.896. As expected, a person with a higher rank has won more rounds, so this is as expected.

Moving on, we can discuss the other model used, the decision tree. The decision tree helps us understand the actual meaning of any relationships between variables and whether or not they are of any real importance. The primary way that we evaluate this is by using the following:

- **Gini Coefficient.** Denoted by it's name, *Gini*, Gini coefficients are used with decision tree algorithms to measure the effectiveness of branches in a data tree. In a good model, we expect these to be closer to zero towards the leaves of a data tree [40].

Gini is calculated as follows: $Gini = 1 - \sum_{k=1}^{n} p^2 k$

Here is a graph that represents the decision tree found within the experiment.

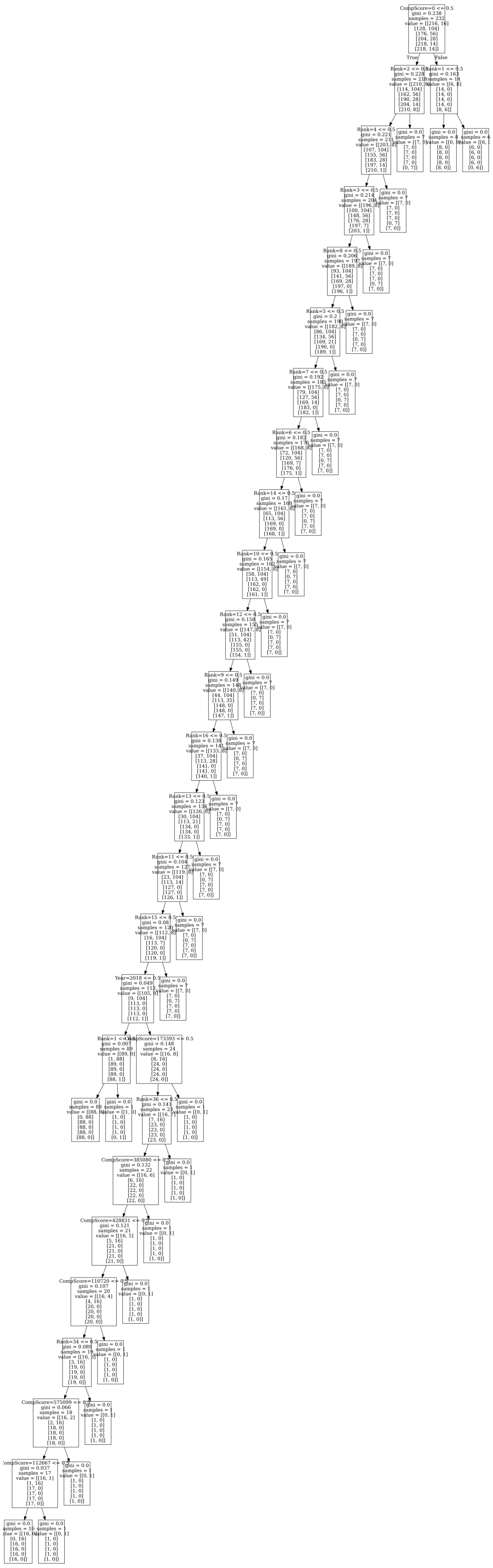

CompScore=0 <= 0.5 gini = 0.238 samples = 232 value = [[216, 16] [128, 104] [176, 56] [204, 28] [218, 14] [218, 14]]
True
False
Rank=2 <= 0.5 gini = 0.228 samples = 218 value = [[210, 8] [114, 104] [162, 56] [190, 28] [204, 14] [210, 8]]
Rank=1 <= 0.5 gini = 0.163 samples = 14 value = [[6, 8] [14, 0] [14, 0] [14, 0] [14, 0] [8, 6]]
Rank=4 <= 0.5 gini = 0.221 samples = 211 value = [[203, 8] [107, 104] [155, 56] [183, 28] [197, 14] [210, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [7, 0] [7, 0] [0, 7]]
gini = 0.0 samples = 8 value = [[0, 8] [8, 0] [8, 0] [8, 0] [8, 0] [8, 0]]
gini = 0.0 samples = 6 value = [[6, 0] [6, 0] [6, 0] [6, 0] [6, 0] [0, 6]]
Rank=3 <= 0.5 gini = 0.214 samples = 204 value = [[196, 8] [100, 104] [148, 56] [176, 28] [197, 7] [203, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [7, 0] [0, 7] [7, 0]]
Rank=8 <= 0.5 gini = 0.206 samples = 197 value = [[189, 8] [93, 104] [141, 56] [169, 28] [197, 0] [196, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [7, 0] [0, 7] [7, 0]]
Rank=5 <= 0.5 gini = 0.2 samples = 190 value = [[182, 8] [86, 104] [134, 56] [169, 21] [190, 0] [189, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [0, 7] [7, 0] [7, 0]]
Rank=7 <= 0.5 gini = 0.192 samples = 183 value = [[175, 8] [79, 104] [127, 56] [169, 14] [183, 0] [182, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [0, 7] [7, 0] [7, 0]]
Rank=6 <= 0.5 gini = 0.182 samples = 176 value = [[168, 8] [72, 104] [120, 56] [169, 7] [176, 0] [175, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [0, 7] [7, 0] [7, 0]]
Rank=14 <= 0.5 gini = 0.17 samples = 169 value = [[161, 8] [65, 104] [113, 56] [169, 0] [169, 0] [168, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [7, 0] [0, 7] [7, 0] [7, 0]]
Rank=10 <= 0.5 gini = 0.165 samples = 162 value = [[154, 8] [58, 104] [113, 49] [162, 0] [162, 0] [161, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=12 <= 0.5 gini = 0.158 samples = 155 value = [[147, 8] [51, 104] [113, 42] [155, 0] [155, 0] [154, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=9 <= 0.5 gini = 0.149 samples = 148 value = [[140, 8] [44, 104] [113, 35] [148, 0] [148, 0] [147, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=16 <= 0.5 gini = 0.138 samples = 141 value = [[133, 8] [37, 104] [113, 28] [141, 0] [141, 0] [140, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=13 <= 0.5 gini = 0.123 samples = 134 value = [[126, 8] [30, 104] [113, 21] [134, 0] [134, 0] [133, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=11 <= 0.5 gini = 0.104 samples = 127 value = [[119, 8] [23, 104] [113, 14] [127, 0] [127, 0] [126, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=15 <= 0.5 gini = 0.08 samples = 120 value = [[112, 8] [16, 104] [113, 7] [120, 0] [120, 0] [119, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Year=2018 <= 0.5 gini = 0.049 samples = 113 value = [[105, 8] [9, 104] [113, 0] [113, 0] [113, 0] [112, 1]]
gini = 0.0 samples = 7 value = [[7, 0] [7, 0] [0, 7] [7, 0] [7, 0] [7, 0]]
Rank=1 <= 0.5 gini = 0.007 samples = 89 value = [[89, 0] [1, 88] [89, 0] [89, 0] [89, 0] [88, 1]]
CompScore=173393 <= 0.5 gini = 0.148 samples = 24 value = [[16, 8] [8, 16] [24, 0] [24, 0] [24, 0] [24, 0]]
gini = 0.0 samples = 88 value = [[88, 0] [0, 88] [88, 0] [88, 0] [88, 0] [88, 0]]
gini = 0.0 samples = 1 value = [[1, 0] [1, 0] [1, 0] [1, 0] [1, 0] [0, 1]]
Rank=36 <= 0.5 gini = 0.141 samples = 23 value = [[16, 7] [7, 16] [23, 0] [23, 0] [23, 0] [23, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
CompScore=385080 <= 0.5 gini = 0.132 samples = 22 value = [[16, 6] [6, 16] [22, 0] [22, 0] [22, 0] [22, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
CompScore=428831 <= 0.5 gini = 0.121 samples = 21 value = [[16, 5] [5, 16] [21, 0] [21, 0] [21, 0] [21, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
CompScore=110720 <= 0.5 gini = 0.107 samples = 20 value = [[16, 4] [4, 16] [20, 0] [20, 0] [20, 0] [20, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
Rank=34 <= 0.5 gini = 0.089 samples = 19 value = [[16, 3] [3, 16] [19, 0] [19, 0] [19, 0] [19, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
CompScore=575099 <= 0.5 gini = 0.066 samples = 18 value = [[16, 2] [2, 16] [18, 0] [18, 0] [18, 0] [18, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
CompScore=112667 <= 0.5 gini = 0.037 samples = 17 value = [[16, 1] [1, 16] [17, 0] [17, 0] [17, 0] [17, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]
gini = 0.0 samples = 16 value = [[16, 0] [0, 16] [16, 0] [16, 0] [16, 0] [16, 0]]
gini = 0.0 samples = 1 value = [[0, 1] [1, 0] [1, 0] [1, 0] [1, 0] [1, 0]]

Figure 9: A more readable image can be found at https://i.imgur.com/mLXv6sU.png

At the root node the Gini coefficient was 0.238 for a sample size of 232 samples. Moving towards the true side of the graph, we found that Gini coefficients gradualy become closer to zero, with last branching node to only contain a Gini coefficient of 0.037. It should be noted that the data tree node is very lopsided. This could possibly indicate that the computational cost per each decision is rather high [40].

5.7 Conclusion

The deductions made from both of these experiments must be approached differently, since they are seperate models. According to the chart, we can see that the *R* squared value for Rank vs. Year became 0.010, for Rank vs. Score it was 0.242, and for Rank vs. Rounds Won it was 0.803. Table 4. This should come as no real suprise, since we expect that any person who won more rounds has a higher rank then the others who did not win. We know that from the rather low values comparatively for Score and Year that there does not seem to be a strong linear relationship between Rank vs. Year and Rank vs. Score.

As for the data tree model, it was discovered that the Gini coefficients for the model were indeed closer to zero but that the data tree itself was heavily lopsided Figure 9. This suggests that although their may be vague linear relationship when classifying the elements of the data tree, it has high cost for it's usage, suggesting it may not fit to be a very useful model. The Gini coefficient at the first branch was 0.238, and the Gini coefficient at the last one was 0.37. Pruning of unnessecary leaves, mainly any branches whose Gini coefficients are already zero, could help this matter [40].

5.8 Specification

```
swagger: "2.0"
info:
  version: "1.0.0"
  description: "Analyzes Tetris games from the Tetris World Championships"
  termsOfService: "http://swagger.io.terms"
  contact:
    name: "Tetris Score Analyzer"
    email: "zwlevy@iu.edu"
  license:
    name: "Apache 2.0"
    url: "http:///www.apache.org/licenses/LICENSE-2.0"
host: "localhost:8080"
basePath: "/cloudmesh/ai/tetris"
```

```
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /lin_reg/download/<output>:
    get:
      operationId: project-code.linear_regressor.download
      produces:
        - "application/json"
      responses:
        "200":
          description: "OK"
          shema: {}

  /lin_reg/read:
    get:
      operationId: project-code.linear_regressor.read_csv
      produces:
        - "application/txt"
      responses:
        "200":
          description: "OK"
          schema: {}

  /lin_reg/analyze:
    get:
      operationId: project-code.linear_regressor.linear_regression
      produces:
        - "application/png"
      responses:
        "200":
          description: "OK"
          schema: {}

  /dt/download/<output>:
    get:
      operationId: project-code.data_tree.download
      produces:
        - "application/json"
      responses:
        "200":
          description: "OK"
          schema: {}

  /dt/analyze:
    get:
      operationId: project-code.data_tree.data_tree
      produces:
        - "application/png"
      responses:
        "200":
          description: "OK"
          schema: {}
```

6 Political Bias and Voting Trends

Mercedes Olson Jarod Saxberg
mercolso@iu.edu jsaxberg@iu.edu
Indiana University
hid: sp19-222-96 sp19-222-100
github: ⌧
code: ⌧

6.1 Abstract

Sklearn's K-Nearest Neighbors algorithm was used to train a classifier which determines if a presidential candidate would win or lose an election based on their support or opposition to ten issues- healthcare, military, education, taxing the wealthy/businesses, women's rights, globalism, gun rights, infrastructure, minority rights, and immigration. Correlations were attempted to be made between the algorithm results and actual presidential election results.

6.2 Introduction

K-Nearest Neighbors (KNN) is a basic classification algorithm. It is non-parametric, so it does not make any underlying assumptions about the distribuion of data. A workflow for a typical KNN algorithim is as follows:

- Store training samples in an array
- Calculate Euclidean distance between the test data point and all training data points
- Make a set of the smallest distances obtained
- Return the majority label from the set

The algorithm finds the training data point that is most similar to the testing data point. Once found, it determines the classification of the test data from the classification from the nearest training data. This was used with a dataset containing ten features that will determine one classification. The features and classification are aligned so that the algorithm will attempt to predict if a candidate would win an election based on their support or opposition of the ten features.

6.3 Requirements

In order to run this project, docker must be installed and working. Once complete, a docker account must be created to use docker. After creating an account, download the Makefile and Dockerfile to a new directory. To build the docker container, run the command `make docker-all`. The Docker container will download the necessary python libraries found within requirements.txt and then start the REST service on its own. The endpoints will then be available to access through a web browser or curl client.

6.4 Design

The design of this project is quite simple for people to use. From a high-level perspective, all a user needs to do is build a docker container and proceed to the endpoints defined. To break this down, the project is split up into three parts: the python code containing the machine learning logic, the REST API connecting the python code to a website that can be navigated to, and a docker container that houses all of the files necessary for the project. Below each part will be discussed in more detail.

6.4.1 Python

This project makes use of sklearn's K-Nearest Neighbors algorithm [41] to perform machine learning on the dataset. Sklearn allows easy use of many machine learning algorithms. Using their K-Nearest Algorithm is as simple as using the lines:

```
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)

pred = classifier.predict(x_test)
```

With the use of that code, everything within the scope of this project can be completed. When users pass their own hypothetical candidate with ten arguments, the arguments are added to an array that is then passed through the predict function and the classifier determines if the candidate would win or lose based on the given arguments.

6.4.2 REST Service

The REST Service allows simple connection between backend python code and a frontend website/API where users can pass data in and receive it back as a json formatted object. People can easily access the classifier through the API without the need to build it and run it on their own. Endpoints are developer-designed urls that can be navigated to so users can run the classifier and other useful features. The basepath is what prefixes all other endpoints; it can be thought of as a top level directory and all of the endpoints are files within the directory structure. The specification in YAML that showcases the basepath and endpoints used can be found in Section 6.10. For readability and length the endpoint that allows a user to perform a custom classification used abbreviations. To clarify the abbreviations the end point is provided next with the abbreviations spelled out.

```
/run/custom/<neighbors>/<healthcare>/<military>/<education>/<tax
wealthy\>/<womens rights>/<globalism>/<gun
rights>/<infrastructure>/<minority rights>/<immigration>
```

6.4.3 Docker

Docker containers were used in this project to create an image that held all of the requirements to run this project. Containers are good to use because it eliminates the requirement of downloading all the necessary files to a personal computer and allows them to be easily removed by deleting the image. It also allows one common operating system–in this case, Ubuntu–to be used across all projects, thus eliminating the troubles that can arise from a mixture of operating systems.

6.5 Dataset

The dataset for this project had to be built from scratch as there were not any easily accessible datasets with the features desired. In order to create the dataset, google sheets was used to input data, and then the spreadsheet was downloaded as a csv file. After converting the dataset to a csv file, it was then uploaded to a website that can be accessed through the python requests library. The information obtained from the Federal Election Commissions [42] website included the top 4 presidential candidates based off the total votes and percentage of votes received from the years 1988 to 2016 with their respective parties. After selecting the candidates ten topics were chosen. The topics include healthcare, military, education, taxing the wealthy, women's rights, globalism, gun rights, infrastructure, minority rights, and immigration. Each presidential candidate received a 1 or 0 if they supported or opposed a topic, based on research from the website OnTheIssues [43] and wikipedia pages about the respective campaigns. An example would be if the candidate had a 0 in the 3rd element and 1 in the 5th element, their actions in the past show opposition to education and support for women's rights. One final thing to take into consideration when making this data is personal bias and the limitations of records can affect the results.

6.6 Results

It is difficult to draw conclusive results from this machine learning application primarily because of the limited size of the dataset. There are only 32 entries in the dataset, so the algorithm does not have a lot of data to learn from. Running the algorithm multiple times returns multiple different values for the precision, recall, and f1-score. It is primarily dependent on which entries are chosen for the testing and training datasets. The algorithm could be improved with additional candidates dating further back from 1988, but the data becomes more sparse pre 1988.

Using the KNN algorithm from sklearn requires an argument for n_neighbors which will affect the accuracy of the algorithm. A simple loop was developed to determine the best n_neighbor argument for each iteration of testing and training datasets. The loop determined the error rate that each possible n_neighbor argument produced, and creates a graph plotting the n_neighbor argument against the error rate. Figure 10 is a sample graph generated by one iteration of the testing and training datasets.

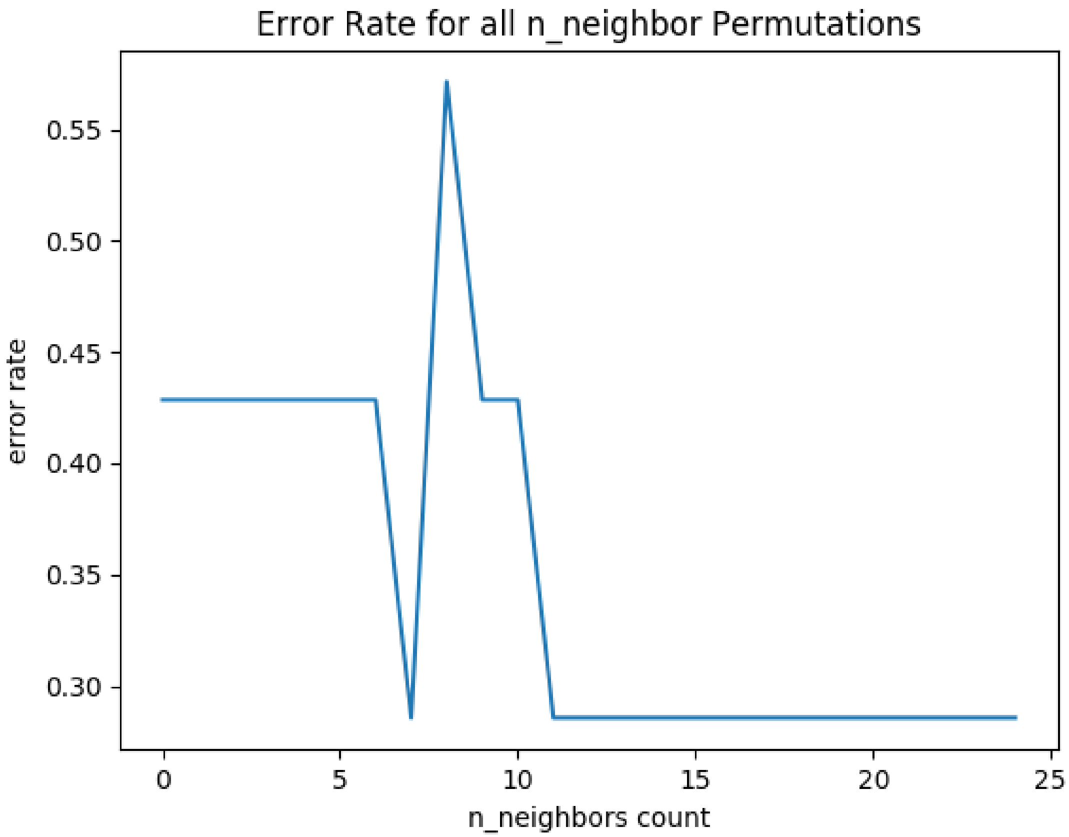


Figure 10: Error Rate vs. Number of Neighbors

6.7 Discussion

Machine learning and artificial intelligence are both hot topics having grown exponentially in popularity during the past ten years–many people are racing to find new ways to apply these algorithms to real-world scenarios. An area of interest for this project is how machine learning applies to politics, specifically data-driven politics. The application has been difficult, however. As shown in [44], it is difficult to draw correlations between analysis results and electoral outcomes. Prior research has been done connecting measures of public opinion measured from polls with sentiment measured from text as shown in [45], and it shows promise having correlations as high as 80%. The research shows how sentiment from text could be used as a substitute for typical polling and predict movement in the polls; however, using Twitter as the source for text nowadays would create issues as the amount of bots on Twitter has grown exponentially since this research was done in 2011. There are other more reliable ways to obtain data for use with machine learning that are not affected in the ways that social media is. Combining typical polling with machine learning could be a path forward that might produce positive results, and help politicians run their campaigns. Machine learning can also help voters make more informed decisions about the candidates available for them to vote on and the stances they hold on certain issues. There are many different applications of machine learning to political data, it is just a matter of finding the best approach and refining the algorithm until error rates are low.

6.8 Conclusion

As stated in the results section, it is difficult to draw conclusions to the accuracy of this application of machine learning because there is only 32 entries total in the dataset. The precision, recall, and f1-scores varied each time the algorithm ran. Likewise, the best n_neighbors argument varied with each testing and training dataset used.

To improve the results, more candidates from past elections could be added–it would require more in depth research on the topics they support as the information is more difficult to access. It would also improve over time as the amount of candidates increases and their campaigns would be better documented. The predictions will gradually get more accurate as time goes on and more data is collected.

6.9 Work Breakdown

Going through candidates and creating the dataset was an even contriubtion from both contributors. The coding aspect was spilt in a way that Jarod set up the KNN algorithm and Mercedes set up the training and testing part of the algorithm. The dockerfile was worked on by both authors, and running and testing the docker containers was done on Mercedes's computer. The report was broken down in segments with Mercedes writing the initial outline and Jarod working on the abstact. Both colaborators worked evenly on writing and editing the rest of the sections of the report.

6.10 Specification

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "presidential support"
  description: "Attempts to determine how much support a candidate will receive
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/cloudmesh/ai/voting"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "application/json"
paths:
  /run/custom/<neighbors>/<hlt>/<mil>/<edu>/<tax>/<wmr>/<glb>/<gnr>/<inf>/<mnr>/
    get:
      tags:
        - RUN_CUSTOM
      operationId: run.run_custom
      description: "Runs an analysis based on given arguments"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Run custom analysis"
          schema: {}
  /run/test:
    get:
      tags:
        - RUN_TEST
      operationId: run.run_test
      description: "Runs an analysis based on test/train data from dataset."
      produces:
        - "application/json"
      responses:
        "200":
          description: "Run test analysis"
          schema: {}
  /run/neighbors:
    get:
      tags:
        - RUN_NEIGHBORS
      operationId: run.neighbors
      description: "Determines best neighbor argument to use for KNN algorithm"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Run neighbor search"
          schema: {}
  /data/download/<output>:
    get:
      tags:
        - DATA
      operationId: data.download
      description: "Downloads data from an external location"
      produces:
        - "application/json"
      responses:
        "200":
          description: "Data info"
          schema: {}
  /data/show/graph:
    get:
      tags:
        - DATA_GRAPH
      operationId: data.graph
      description: "Shows the graph generated from the neighbors endpoint"
      produces:
        - "application/png"
      responses:
        "200":
          description: "Show graph"
          schema: {}
```

7 Spam Analysis with Spamalot

Eric Bower, Tyler Zhang
epbower@iu.edu, tjzhang@iu.edu
Indiana University Bloomington
hid: sp19-222-101
Github: ☒
code: ☒

Keywords: Spam

7.1 Abstract

Spam emails are an issue in cybersecurity because they can contain phishing scams or malware that can steal private information from the user, which can often lead to identity theft. These emails are particularly dangerous because many are seemingly innocuous. We are exploring the possibility of predicting malicious intent in emails by using a machine learning algorithm. We chose to use the Support Vector Machines (SVM) algorithm to classify spam emails due to its superior classification performance over another commonly used classification model, the Naive Bayes algorithm. Successfully classifying a malicious email can prevent harmful situations for users.

7.2 Introduction

The goal is the creation of a service that can classify spam emails. By the term "spam email", we refer to emails that are sent with malicious intent against the recipient. Emails without malicious intent will be referred to as a "ham email". Common forms of malicious intent that steal or restrict a user's personal data include phishing and malware.

Phishing describes a process in which an attacker impersonates a trustworthy third party in an attempt to obtain sensitive information [46]. In a recent phishing attack, a link was sent to Snapchat users telling them to enable two-step authentication. This link collected the login information of more than 50,000 Snapchat users. Users were under the impression that they were making their private information more secure, but they instead exposed their personal information to hackers, demonstrating how phishing attacks can be particularly deceptive.

Spam emails can also have dangerous attachments that contain malware. When these attachments are downloaded, the malware is unloaded onto the computer system. A common example of malware is called ransomware, which encrypts local files and network files, effectively preventing the user from accessing their own files. The ransomware will ask the user to pay in exchange for decrypting their files. Even if the user pays, the attacker still has control over the user's data, which may lead to identity theft [47].

An email classifier can prevent users from being hit by phishing scams and malware. We used a machine learning algorithm to classify malicious emails. An important consideration for the user is to minimize misclassification errors. When creating our service, we need to be careful to minimize the number of spam emails that are erroneously labeled as ham and the number of ham emails that are erroneously labeled as spam. If misclassification occurs, the user is at risk of opening dangerous spam emails and missing important ones.

7.3 The Algorithm

We considered the Naive Bayes and Support Vector Machine (SVM) algorithms for our implementation of spam classification. Naive Bayes and SVM are both supervised learning algorithms, which means that they are trained with data that is already labeled. Both algorithms have strengths and weaknesses. The Naive Bayes algorithm generally is faster and less computationally complex [48]. SVM is slower than Naive Bayes but typically tends to be more statistically robust [49]. We experimented with both algorithms and chose which one to use based on their performance and respective statistics.

7.3.1 Naive Bayes

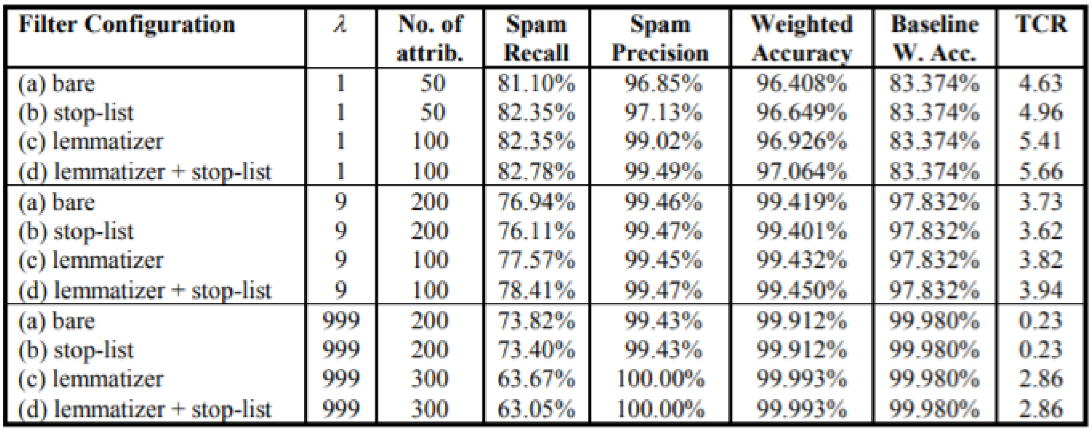
Naive Bayes classifiers are very commonly used for spam filtering and document classification problems [50]. The Naive Bayes algorithm relies on Bayes' probability theorem, which expresses a relationship between the probability of the occurrence of an event c given the occurrence of other events, x_1 through x_n [48]. Representing E as $(x_1, x_2, ...x_n)$, the probability of an event c given E is given in Equation 5 from [48].

$$P(c|E) = \frac{P(E|c)P(c)}{P(E)} \tag{5}$$

In terms of classification, the vector E would be the features of the data point, and c is the classification of that data point (either ham or spam). To create a binary classifier, with two classifications being $c = spam$ and $c = ham$, the classification of a data point with a feature vector E is given in Equation 6 from [48].

$$f_b(E) = \frac{P(c = spam|E)}{P(c = ham|E)} >= 1 \tag{6}$$

The terms $P(c = spam|E)$ and $P(c = ham|E)$ are both calculated using Equation 5. The classification is spam if $f_b(E)$ is greater than or equal to one, and ham if $f_b(E)$ is less than one. Other researchers have found that the Naive Bayes classification model has decent precision and recall values when classifying spam emails. The following Figure 11 shows the results of such an experiment of a Naive Bayes classifier run on a data set of 2893 total messages:



Filter Configuration	λ	No. of attrib.	Spam Recall	Spam Precision	Weighted Accuracy	Baseline W. Acc.	TCR
(a) bare	1	50	81.10%	96.85%	96.408%	83.374%	4.63
(b) stop-list	1	50	82.35%	97.13%	96.649%	83.374%	4.96
(c) lemmatizer	1	100	82.35%	99.02%	96.926%	83.374%	5.41
(d) lemmatizer + stop-list	1	100	82.78%	99.49%	97.064%	83.374%	5.66
(a) bare	9	200	76.94%	99.46%	99.419%	97.832%	3.73
(b) stop-list	9	200	76.11%	99.47%	99.401%	97.832%	3.62
(c) lemmatizer	9	100	77.57%	99.45%	99.432%	97.832%	3.82
(d) lemmatizer + stop-list	9	100	78.41%	99.47%	99.450%	97.832%	3.94
(a) bare	999	200	73.82%	99.43%	99.912%	99.980%	0.23
(b) stop-list	999	200	73.40%	99.43%	99.912%	99.980%	0.23
(c) lemmatizer	999	300	63.67%	100.00%	99.993%	99.980%	2.86
(d) lemmatizer + stop-list	999	300	63.05%	100.00%	99.993%	99.980%	2.86

Figure 11: NB Ling-Spam Results[51]

Metrics In Figure 11, two important performance metrics of the Naive Bayes algorithm were recall and precision, which are both standard metrics used to evaluate the effectiveness of a model. Recall is defined as the number of true positives divided by the sum of true positives and false negatives. To put that definition in the context of a spam classifier, recall measures the ratio of spam emails correctly identified as spam compared to the number of actual spam emails in the data set.

Precision is defined as the number of true positives divided by the sum of true positives and false positives. To put that definition in the context of a spam classifier, precision measures the ratio of spam emails correctly identified as spam compared to the number of emails the classifier thinks are spam. For both metrics, higher percentages show a superior model. The figure Figure 11 demonstrates decently high percentages for recall and precision when using Naive Bayes as a spam email classifier.

7.3.2 Support Vector Machines (SVM)

SVM is also used to work in text classification [50]. SVM models construct hyper-planes in the feature space of the dataset which can be used for classification. The hyperplane is chosen by finding the optimal plane that maximizes its margins of separation between points of all classes [52]. In other words, data points are separated from the others based on their features in an optimal manner.

To better illustrate how SVM works, one can imagine all points of a data set plotted based on their attributes. The data points that belong to a certain classification will generally be plotted together in regions because they normally have similar attributes. A hyperplane is an imaginary divider between these classification regions that is mathematically calculated based on distance. The following figure Figure 12 shows a visualization of a two-dimensional hyperplane:

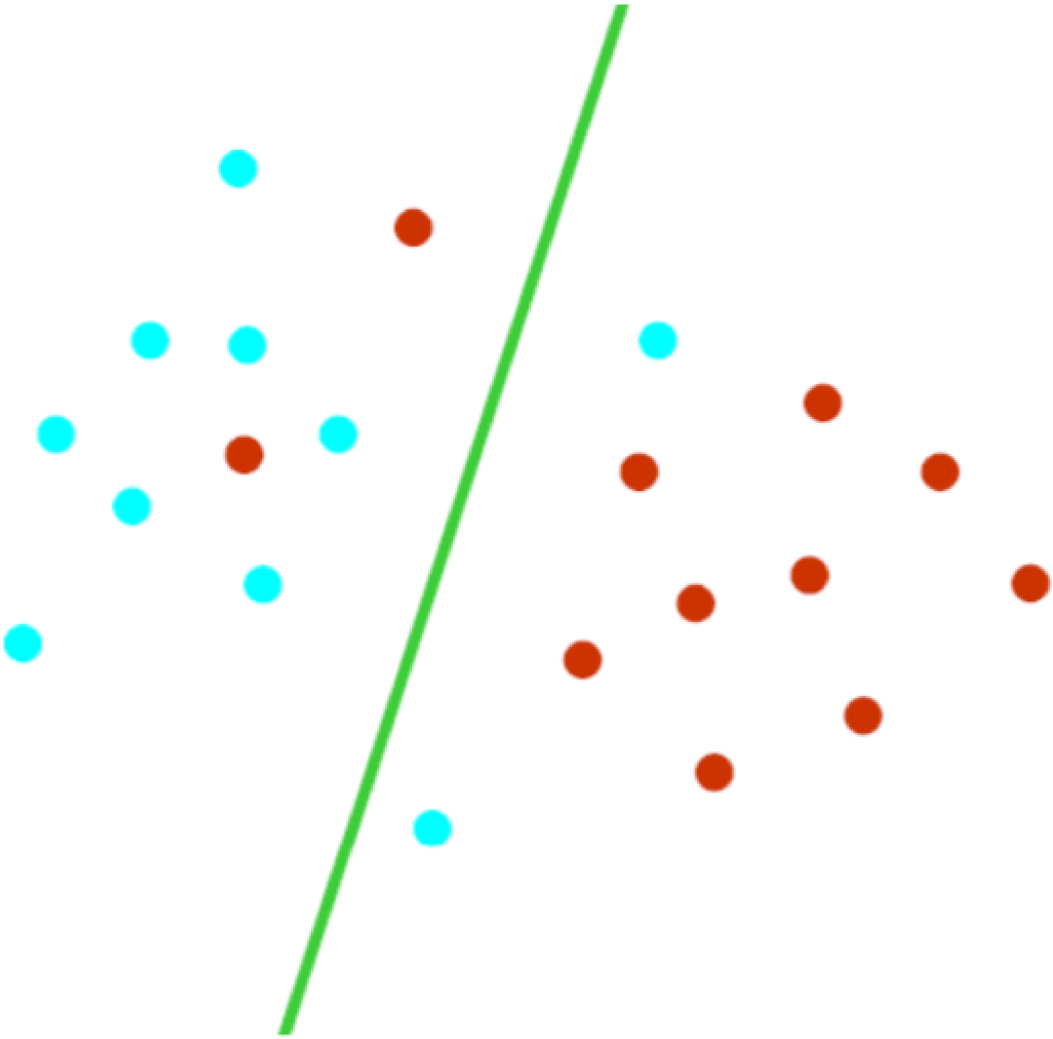


Figure 12: SVM 2D Visualization[52]

This figure shows a divider between general regions of points classified as blue and points classified as red. Notice that the hyperplane is imperfect; there are some red points on the blue side of the hyperplane, and there are some blue points on the red side. Such error is inevitable, but it can be minimized mathematically. To make a classification on a new piece of data, the algorithm plots the attributes of a new point and makes a prediction based on which side of the hyperplane the point falls on.

SVM algorithms are very effective classifiers when working with datasets that utilize a large number of features [52]. In the case of emails, we can calculate the frequency of words contained in the email, which will be a vector with a large number of features [50]. As a result, SVM is an appropriate model to classify spam emails due to the high dimensional dataset.

7.4 The Data Set

The data set used to train the machine learning algorithm is taken from a project that tested the effectiveness of five different variations of Naive Bayesian classifiers on classifying spam emails. It contains the text of ham and spam messages from a member of Enron corpus with random ham-spam ratio [53]. These emails are all labeled as ham or spam.

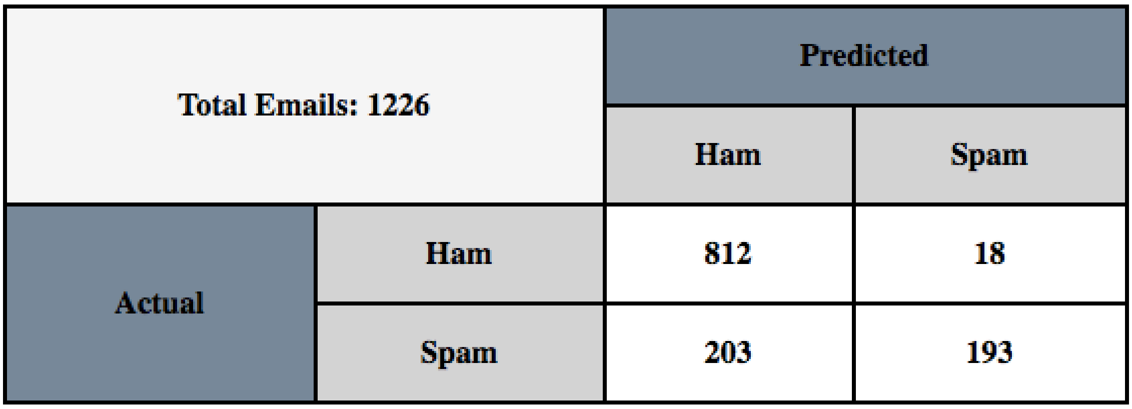
The raw messages of each email generally contain too much information to be considered useful to train the classifier. In particular, raw email message text has many characters that contribute little to the classification of the email. To counteract this, we removed all non-alphabetical characters from the data set. Also, we removed any words that were only one character long, such as “a” and “I”. Finally, we removed all instances of the word “the”, which turned out to be one of the most common words in all the training emails.

It has been shown that the process of lemmatization improves classification performance for spam filtering [51]. Lemmatization is the process of grouping together variations of the same root word. For instance, a lemmatizer would group all instances of the words “include”, “includes”, and “included” in the same category. The data set that we used to train our algorithm was already lemmatized, so we did not need to go through this process ourselves.

Once the email texts were filtered, we then created vectors of word frequencies from each email. These are treated as the features for the machine learning model. Our dataset was already labeled as “spam” or “ham”, making it possible to train the SVM and Naive Bayes supervised learning models. We shuffled the list of word frequency vectors from the dataset and randomly chose 80% of these vectors to train both the SVM and Naive Bayes models. We used the remaining 20% of these vectors to assess the quality of each model.

7.5 Model Results

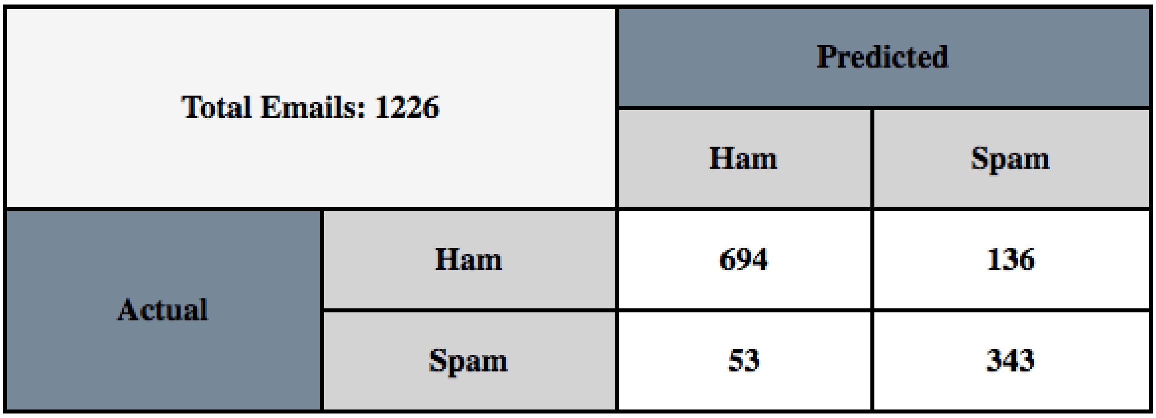
The confusion matrix generated from the Naive Bayes algorithm is represented in Figure 13.



Total Emails: 1226		Predicted	
		Ham	Spam
Actual	Ham	812	18
	Spam	203	193

Figure 13: NB Confusion Matrix

The confusion matrix generated from the SVM algorithm is represented in Figure 14.



Total Emails: 1226		Predicted	
		Ham	Spam
Actual	Ham	694	136
	Spam	53	343

Figure 14: SVM Confusion Matrix

We can use the above confusion matrices to obtain metrics for each model. Recall measures the ratio of spam emails correctly identified as spam compared to the number of actual spam emails in the data set. For the Naive Bayes model, this value is $\frac{193}{203+193} = 0.487$. For the SVM model, this value is $\frac{343}{53+343} = 0.866$.

Precision measures the ratio of spam emails correctly identified as spam compared to the number of emails the classifier thinks are spam. For the Naive Bayes model, this value is $\frac{193}{18+193} = 0.915$. For the SVM model, this value is $\frac{343}{136+343} = 0.716$.

Although the Naive Bayes method has higher precision than the SVM, it has significantly lower recall.

The impact this would have on the user is dangerous: the confusion matrix Figure 13 shows that the Naive Bayes model flags a great number of spam emails as ham, exposing the user to more malicious emails. On the other hand, the SVM method has a significantly higher recall value, but at a lower precision value. The consequences of this are not as dire for the user: the confusion matrix Figure 14 shows the SVM model flags more ham emails as spam. While this is inconvenient for the user, the SVM method is safer because it would expose fewer malicious emails to the user, at the cost of flagging benign emails as spam more often. For this reason, we decided to use the SVM model for our classification method.

7.6 Implementation

7.6.1 The Server

We used Swagger OpenAPI to create our server. This package lets us conveniently define our server's endpoints and REST API operations in a concise YAML file. In addition to Swagger OpenAPI, we also greatly utilized the Flask package. In our YAML file, our server has one endpoint called "upload", which acts as a REST POST operation. The upload() function corresponding to this endpoint is defined in a file called gatherData.py. From the Flask package, upload() uses Flask.request() to read in a user's text file. Once the file has been accepted, the text file is processed and the classification occurs. The results of the classification are then returned and displayed to the user as an HTML page using Flask.render_template(). The upload() function and the classification process is explained in more detail in the next section.

It should be noted that the "upload" endpoint cannot be reached directly by URL; it can only be accessed by pressing a JavaScript button on the home page of the server, which then calls the Flask.request() function and starts the classification process. If a user attempts to access the "upload" endpoint via URL instead of clicking the button, they will be taken to an error page. Once a user successfully sees the classification results, they are given the option to return to the home page via another JavaScript button, where they can upload another email for classification.

We created a Dockerfile and a Makefile that contains all the necessary commands needed to host the server. This Dockerfile runs Ubuntu and runs commands to clone files from our Github to get the service running. Running the server in a container such as Docker is beneficial because the installation will not interfere with the host system, and the container is easy to remove once installed.

The following diagram Figure 15 shows the basic workflow of our server:

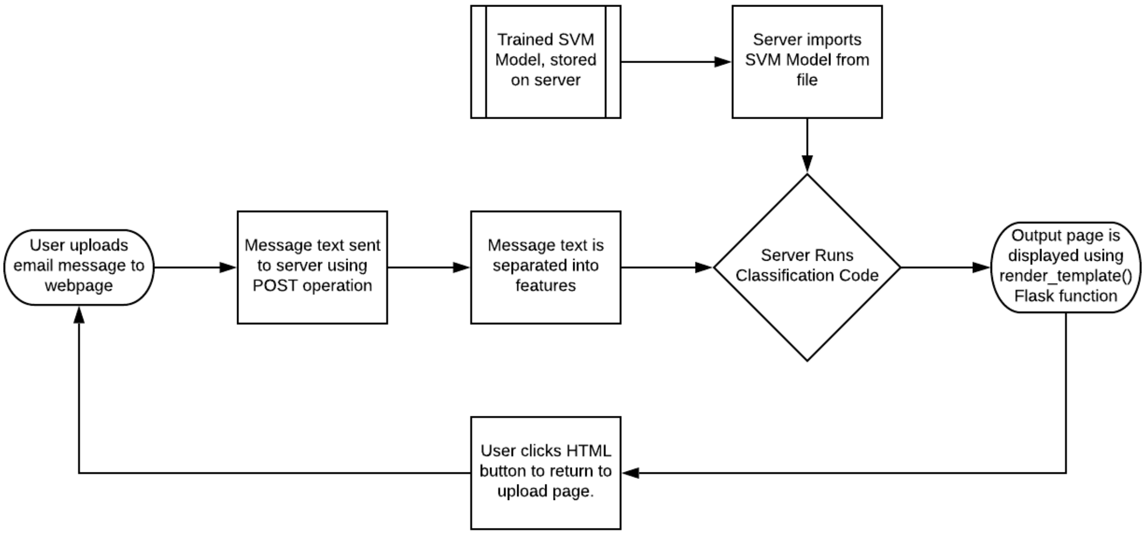

Trained SVM Model, stored on server
Server imports SVM Model from file
User uploads email message to webpage
Message text sent to server using POST operation
Message text is separated into features
Server Runs Classification Code
Output page is displayed using render_template() Flask function
User clicks HTML button to return to upload page.

Figure 15: Classification Flowchart

7.6.2 The Upload Function and Classification

The upload() function handles file uploading and makes other function calls to perform classification. The file is first retrieved by utilizing the Flask.request() function from the Flask package. This file must have a ".txt" extension and should contain the body of the email to be classified.

Once the file is uploaded, upload() sends the file to a function called extract_features(). The extract_features() function converts the email text into a word frequency vector. This vector is treated as the features of the email and is used as an input for the SVM model to classify the email as spam or ham.

Our server contains a previously-trained SVM model file, so it will not need to retrain the model each time a user uploads a new file. We used the sklearn package to train an SVM model on a large number of emails, creating a large database of word frequency vectors and their corresponding labels. Using the pickle package, we saved this model as a file on the server. When the extract_features() function successfully returns a feature vector for the user's uploaded email, the server uses the pickle package to load the saved SVM model and labels the user's email as ham or spam.

The final pieces of information that the upload() function calculates are performance statistics for the SVM model to create a confusion matrix, from which the user can calculate model metrics such as precision and recall. This gives the user insight on the performance of the SVM model. Once the server has made a prediction and has performance statistics, it calls Flask.render_template() to display an HTML file with the returned variables. This is the page the user sees after uploading their email file.

7.6.3 Specification

```
swagger: "2.0"
info:
  version: "0.0.1"
  title: "spamInfo"
  description: "An intelligent system to determine whether a given email is spam
  contact:
    name: "Spamalot"
  license:
    name: "Apache"
host: "localhost:8080"
basePath: "/"
schemes:
  - "http"
consumes:
  - "application/json"
produces:
  - "text/html"
paths:
  /upload:
    post:
      operationId: py_scripts.gatherData.upload
      description: "Uploads a file"
      responses:
        "201":
          description: "Upload"
```

7.7 Conclusion

Our implementation of a spam email classifier is a building block for a more sophisticated spam filter. Currently, our service is only capable of making predictions on one email at a time, and each prediction requires the user to upload a new file containing the email text. A more sophisticated spam filter would be integrated into an email app and could automatically classify incoming emails as spam. Additionally, our service's classification ability is imperfect. As seen in Figure 11, other researchers achieve superior classification performance using Naive Bayes, so there is room for improvement.

In its current state, our implementation is best suited for single users. But with some further development, such as adding the ability to classify multiple emails at once and improving our service's classification performance, a spam filter would be a useful tool for company employees. Our service could prevent employees from opening dangerous emails that could steal company information.

8 REFERENCES

The following refernces are collected automatically from multiple sources.

[1] P. Gouras, “The role of s-cones in human vision,” *Documenta ophthalmologica*, vol. 106, no. 1, pp. 5–11, 2003.

[2] “Rods and cones.” Website [Online]. Available: https://www.cis.rit.edu/people/faculty/montag/vandplite/pages/chap_9/ch9p1.html

[3] D. Mustafi, A. H. Engel, and K. Palczewski, “Structure of cone photoreceptors,” *Progress in retinal and eye research*, vol. 28, no. 4, pp. 289–302, 2009.

[4] M. Simoneau and J. Price, “Neural networks provide solutions to real-world problems: Powerful new algorithms to explore, classify, and identify patterns in data.” Website, 1998 [Online]. Available: https://www.mathworks.com/company/newsletters/articles/neural-networks-provide-solutions-to-real-world-problems-powerful-new-algorithms-to-explore-classify-and-identify-patterns-in-data.html

[5] F. Manessi and A. Rozza, “Learning combinations of activation functions,” *CoRR*, vol. abs/1801.09403, 2018 [Online]. Available: http://arxiv.org/abs/1801.09403

[6] “Python rest apis with flask, connexion, and sqlalchemy.” Website, 2018 [Online]. Available: https://realpython.com/flask-connexion-rest-api/

[7] J. Drayer, S. L. Shapiro, B. Dwyer, A. L. Morse, and J. White, “The effects of fantasy football participation on nfl consumption: A qualitative analysis,” *Sport Management Review*, vol. 13, no. 2, pp. 129–141, 2010.

[8] B. Gagnon, “What is the most important combine event for each nfl position?” 2018 [Online]. Available: https://bleacherreport.com/articles/2760924-what-is-the-most-important-combine-event-for-each-nfl-position#slide6

[9] T. Srivastava, “Introduction to k-nearest neighbors: Simplified (with implementation in python),” 2018 [Online]. Available: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[10] C. Pautasso, “RESTful web service composition with bpel for rest,” *Data & Knowledge Engineering*, vol. 68, no. 9, pp. 851–866, 2009.

[11] E. Novoseltseva, “Top 10 benefits of docker,” 2017 [Online]. Available: https://dzone.com/articles/top-10-benefits-of-using-docker

[12] SPORTTECHIE, “Sensors are taking over sports,” 2016.

[13] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[14] I. K. Fodor, “A survey of dimension reduction techniques,” Lawrence Livermore National Lab., CA (US), 2002.

[15] Available: http://www.fon.hum.uva.nl/praat/manual/k-means_clustering_1__How_does_k-means_clustering_work_.html

[16] *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html

[17] *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html#sklearn.metrics.homogeneity_score

[18] “Harmonic mean,” *Wikipedia*. Wikimedia Foundation, Mar-2019 [Online]. Available: https://en.wikipedia.org/wiki/Harmonic_mean

[19] *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_completeness_v_measure.html

[20] “Kmeans clustering and 3D plotting,” *Kaggle*. [Online]. Available: https://www.kaggle.com/timi01/k-means-clustering-and-3d-plotting

[21] K. Nishida, “Visualizing k-means clustering results to understand the characteristics of clusters better,” *learn data science*. learn data science, Nov-2018 [Online]. Available: https://blog.exploratory.io/visualizing-k-means-clustering-results-to-understand-the-characteristics-of-clusters-better-b0226fb3dd10

[22] G. Seif, “5 quick and easy data visualizations in python with code,” *Towards Data Science*. Towards Data Science, Mar-2018 [Online]. Available: https://towardsdatascience.com/5-quick-and-easy-data-visualizations-in-python-with-code-a2284bae952f

[23] N. Sharma, “Ways to detect and remove the outliers,” *Towards Data Science*. Towards Data Science, May-2018 [Online]. Available: https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba

[24] “Multiclass classification,” *Wikipedia*. Wikimedia Foundation, Apr-2019 [Online]. Available: https://en.wikipedia.org/wiki/Multiclass_classification

[25] AbhishekAbhishek, “Training logistic regression using scikit learn for multi-class classification,” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/32074630/training-logistic-regression-using-scikit-learn-for-multi-class-classification

[26] “Decision tree learning,” *Wikipedia*. Wikimedia Foundation, Apr-2019 [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning

[27] “K-nearest neighbors algorithm,” *Wikipedia*. Wikimedia Foundation, Apr-2019 [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[28] Available: https://multivariatestatsjl.readthedocs.io/en/latest/mclda.html

[29] “Linear discriminant analysis,” *Wikipedia*. Wikimedia Foundation, Apr-2019 [Online]. Available: https://en.wikipedia.org/wiki/Linear_discriminant_analysis#Linear_discriminant_in_high_dimension

[30] “Naive bayes for machine learning,” *Machine Learning Mastery*. Sep-2016 [Online]. Available: https://machinelearningmastery.com/naive-bayes-for-machine-learning/

[31] “1.9. Naive bayes,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html

[32] *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[33] “SVM multi-class classification,” *Apache Ignite Documentation*. [Online]. Available: https://apacheignite.readme.io/docs/svm-multi-class-classification

[34] “Sklearn.svm.SVC,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[35] M. Stickgold Malia, “Replaying the game: Hypnagogic images in normals and amnesics,” Oct. 2000 [Online]. Available: https://science.sciencemag.org/content/sci/290/5490/350.full.pdf

[36] B. News, “Tetris ’helps to reduce trauma’.” Web Page, Jan-2007 [Online]. Available: http://news.bbc.co.uk/2/hi/health/7813637.stm

[37] C. T. W. Championship, “Results.” Web Page, 2018 [Online]. Available: https://thectwc.com/results/

[38] P. S. University, “Correlation coefficient r.” Web Page, 2018 [Online]. Available: https://newonlinecourses.science.psu.edu/stat462/node/96/

[39] Macklin, “Regression performance metrics.” Web Page; Indiana University School of Informatics, Computing, & Engineering, Sep-2018.

[40] Macklin, “Introduction to decision trees.” Web Page; Indiana University School of Informatics, Computing, & Engineering, Oct-2018.

[41] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[42] F. E. Commission, “Election results.” https://transition.fec.gov/pubrec/electionresults.shtml, 2019.

[43] S. M. Group, “Candidates on the issues.” https://www.ontheissues.org/default.htm, 2018.

[44] D. Gayo-Avello, P. T. Metaxas, and E. Mustafaraj, “Limits of electoral predictions using twitter,” in *Fifth international aaai conference on weblogs and social media*, 2011.

[45] B. O’Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, “From tweets to polls: Linking text sentiment to public opinion time series,” in *Fourth international aaai conference on weblogs and social media*, 2010.

[46] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, “Social phishing,” *Communications of the ACM*, vol. 50, no. 10, pp. 94–100, 2007.

[47] L. Bridges, “The changing face of malware,” *Network Security*, vol. 2008, no. 1, pp. 17–20, 2008.

[48] H. Zhang, “The optimality of naive bayes,” *A A*, vol. 1, no. 2, p. 3, 2004.

[49] D. Sculley and G. M. Wachman, “Relaxed online svms for spam filtering,” in *Proceedings of the 30th annual international acm sigir conference on research and development in information retrieval*, 2007, pp. 415–422.

[50] A. Khorsi, “An overview of content-based spam filtering techniques,” *Informatica*, vol. 31, no. 3, 2007.

[51] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, “An evaluation of naive bayesian anti-spam filtering,” *arXiv preprint cs/0006013*, 2000.

[52] S. R. Gunn and others, “Support vector machines for classification and regression,” *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998.

[53] V. Metsis, I. Androutsopoulos, and G. Paliouras, “Spam filtering with naive bayes-which naive bayes?” in *CEAS*, 2006, vol. 17, pp. 28–69.