Siri As a Personalized Food Recommender

Anh Dao
Computer Science Department
Worcester Polytechnic Institute

***Abstract*—Siri the first invented Intelligent Personal Assistants by Apple. However, the use of Siri is not high. Part of it is due to the limitation of its ability to understand users. In this research, we are applying three supervised learning classifiers: Support Vector Machine (SVM), Random Forest and Naive Bayes (NB) to turn Siri into a learning agent so that it could make personalized food recommendations. While RF has the best performance and NB has the worst after our testing, there were various errors in our models and limitations we faced that allows future improvements on this project.**

Keywords—Siri, Support Vector Machine, Random Forest, Naive Bayes

I. INTRODUCTION

Virtual Assistants (VAs) or Intelligent Personal Assistants (IPAs) are software agents that help user perform tasks or services [1]. Siri was the first modern IPA created and implemented on smartphones in 2011 by Apple. Siri helps users to complete some basic tasks like calling someone or setting alarms just by talking to it. The invention of Siri took other tech companies years to catch up [2]. Amazon's Alexa was released in 2014 and Google Assistant was released in 2016 [2]. However, the creation of other IPAs has made Siri lose the race since the later IPAs have a wider range of features [2]. According to the article, Apple has not improved significantly.

Statistically, 98% of iPhone users have tried Siri but 70% of them sometimes or rarely use it [19]. Another report also claims that the number of Siri users dropped by half, from 21% to 11%, in 2017 after the growth of Alexa and Google Assistant [20]. After doing research, we realized that the AI implemented Siri is limited to speech recognition. What Siri does is listening to the voice, converting speech to computer language, data searching and responding [4]. According to [5], the interaction with Siri isn't actual conversations, "but the interactions are normally single-turn inputs which are dealt with independently." There is no back and forth conversation. Siri does not a learning agent. We want to fully develop Siri so that it can become a learning agent: learning from experience.

In this project, we want Siri to be able to access the search history from the phone browser, or Safari, and see what sort of food or entertainment users like in order to make future recommendations based on user's preference especially when user travels to another area. Since this is a broad approach, we are narrowing it down to focus on food recommendation based on user's interest.

The main challenge of this project is that there is no Safari browsing dataset and there was not any similar work done on Siri. To alleviate this, we used a restaurant data with consumer rating, where rating is the iPhone user's interest level based on different restaurant characteristics.

As Siri can make personalized recommendations, the use of Siri will be improved because users won't have to go on Yelp or TripAdvisor to find restaurants when they travel and read the comments on the food/service to see if it's what they are looking for. And since all iPhone users have access to Siri, it will be more convenient to use instead of spending extra on Alexa or Google Assistant.

II. BACKGROUND

Siri uses Natural Language Processing to translate spoken language into text that machines can understand [6]. You might ask "Do I need an umbrella today?" and Siri can understand that you're asking for a prediction of rain [6]. Siri can do it because of the algorithm. It takes in the questions and searches for keywords.

A. Related Works

Even though there is not a significant amount of work done on improving any other aspect of Siri besides speech recognition, there is various work done on predicting human behavior.

SVM method is used in various research to make recommendations based on user preferences. [8] uses SVM to predict movie rating based on movie content and user interests. The model is created based on user's rating, movie information, user's demographic. The research proved that SVM is the best machine learning approach among others such as Linear Classifier, Bayesian learning because the solution identified with SVM is optimized and has a strong generalization ability [9]. One of the advantages of SVM is its accuracy [10] but the choice of parameters impacts on its prediction accuracy [9]. In addition, SVM doesn't suit large dataset since the training time will be high [10].

Random Forest [RF] is also found used in making recommendations based on user's preferences. The study in [12] uses RF to make music recommendations based on user's preferences. The program prompts the user with a minimum of 10 songs and has he/she rate the songs then used the rated data for training. Random Forests is a combination of many decision trees which increases accuracy from multiple suggestions (each suggestion is a tree). RF for said to be better in classification problems than SVM with the dismissal of having to tune parameters; RF is faster and scalable [13]. However, the trade-off for RF is its difficulty on interpretation for human [13].

Naive Bayes (NB) classifier is said to perform well with multiclass classification and it's often used to text classification and recommendation system [20]. A study has

done by Koji Miyahara† and Michael J. Pazzani in UC Irvine using NB algorithm for books, CDs, … recommendations using NB algorithm [18]. The study uses user preferences as features to determine if the book will be liked or disliked. NB it's used for its simplicity. It requires less training which results in a faster process. However, the strong assumption of feature independence is one of its disadvantages. In addition, if a class label has no occurrence, the probability will be zero. This problem is mitigated by Laplace smoothing (explained in the Method section).

In this Project, we will develop Siri into a learning agent. Based on our understanding, we will program all of the above algorithms and compare and contrast the performances with three performance metrics.

B. Paper Organization

We structure the paper in the following order. Section I introduces the problem, its challenges, and benefits of the study. Section II addresses different state-of-art methods that were done in previous similar work and their advantages as well as disadvantages. Section III we describe in details how we will use the algorithms to run the data and the metrics we use to evaluate the algorithm. Section VI displays the results and analysis of the performance of each method. And finally, we will conclude our work on section V.

III. METHODS

In this section, we will discuss step by step how our team approached this problem. Our goal is to identify the best state-of-art method to train Siri on predicting user's interest in restaurants based on different restaurant characteristics and user's preferences, and what features would be likely to impact user's interest. To achieve this goal, we divided our process into three steps (displayed in Figure 1).

Our first step is processing data. We merged, cleaned and sectioned our data based on their importance ranking. To find out the ranks, we ran a built-in function from sklearn. Then we structured four variations of our dataset with more important or less important features.

Our second step is training the sets with three supervised learning algorithms: Support Vector Machine, Random Forest, and Naive Bayes. We applied cross-validation on training data. 80% of our data is alternately trained for ten iterations.

Our final step is evaluating the performance of the three methods. We ran 20% of our data on training models for testing. Then we compared their f1 scores and their learning curves for evaluations.

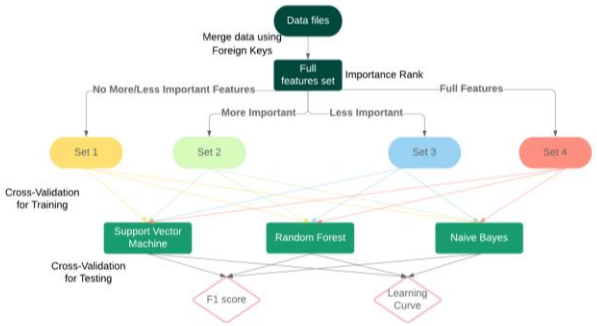


Figure 1. Computational Pipeline of our Methodology. First we merged and cleaned our data. Then we sectioned it into four variations. We tested and trained our models with SVM, RF, and NB models. Finally we evaluated the results with f1 score and learning curve.

A. Datasets

The dataset we used in this research is the Restaurant Data with Consumer Ratings from Kaggle [14]. This data consists of nine files :

Restaurants
1 chefmozaccepts.csv
2 chefmozcuisine.csv
3 chefmozhours4.csv
4 chefmozparking.csv
5 geoplaces2.csv

Customers:
6 usercuisine.csv
7 userpayment.csv
8 userprofile.csv

User-Item-Rating:
9 rating_final.csv

The dataset has over 2300 restaurants and over 130 user profiles. We merged the data among files by using foreign keys which are restaurant ID and customer ID, hot-coded nominal data into binary and cleaned up NaN values. After finishing the merging and cleaning data, our final dataset has 10,000 points. The reason for the expansion from 2000 restaurants and 130 users is due to the multiple ratings from a single user.

B. Sectioning Data

Since we have 27 features, we want to section our main dataset into 3 sets with more and less important features added. To determine the impact of features on our prediction model, we will run ExtraTreesClassifier built-in function from sklearn library to sort out six least important features and six most important features to see how they impact the prediction of our models. In this study, we will test in total four variations of our dataset:

Variation 1: dataset without 6 most and 6 least important features

Variation 2: dataset 1 with 6 most important features added

Variation 3: dataset 1 with 6 least important features added

Variation 4: original dataset with full features

C. Support Vector Machine

Support Vector Machine (SVM) is a supervised learning algorithm and the first algorithm we use to determine user's level of interest on different restaurant features. SVM uses a hyperplane to classify data into two classes. The more features we have, the more dimensions added to the problem. This technique is called "kernelling", mapping data to a higher dimension.

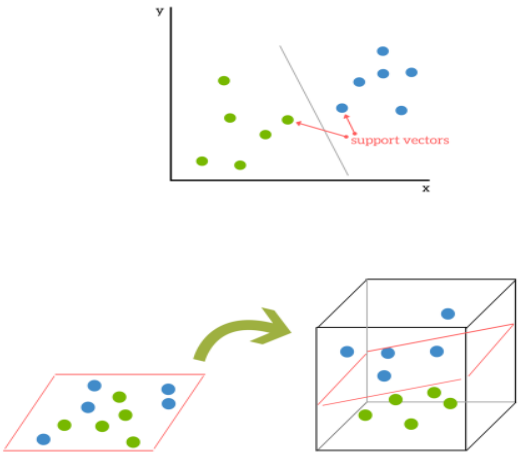


Figure 2. Hyperplane in Two-dimensional and Three-dimensional View. This figure is adapted from [10]

In addition, SVM is inherently a two-class classifier, we have three classes. The most common techniques are one-versus-rest (ovr) classifier and one-versus-one (ovo) classifier. In this study, we are using SVC function from scikit learn library of Python. This function supports two-class classifiers and multiclass classifier and mitigates the problem with kernelling.

There are a few parameters in SVC functions to notice: C, gamma and kernel. C is how much we want to penalize misclassified data points. Lower C means simple and soft margin but underfitting; higher C means less mistake but overfitting. Gamma is model complexity. Lower gamma means less complexity and higher gamma means more complexity (the higher gamma, the more likely the model will classify data point). There are three kernel types: Linear, RBF, Poly. We used Linear kernel for simplicity, set C = 0.1, gamma = auto and 'ovo' classier in SVC built-in function in sklearn.

D. Random Forest

Random Forest (RF) is the second supervised learning classifier we use to test the data. RF algorithm builds multiple Decision Trees and merges them together for a more accurate prediction [17]. RF is an easy method without tuning parameters. RF allows you to measure the importance level of each feature and drop the least important one since the higher number of features will likely lead to overfitting.

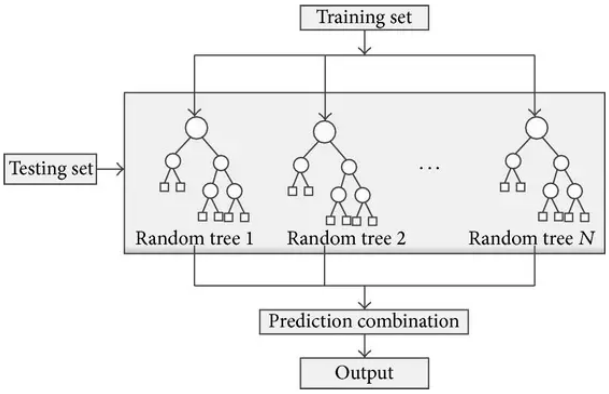


Figure 3. Random Forest Scheme. This figure is adapted from [21].

In this process, we use sklearn built-in Random Forest function. Some critical parameters of the functions for accuracy are *n_estimators* parameter - the number of trees the algorithm will build, *max_features* - maximum features in one tree. As mentioned, the higher the number of trees, the better the model performances, however, the slower the computation. We set *n_estimators* to 250 trees and default *max_features* which is the square root of the data features.

E. Naive Bayes

Naive Bayes is the final supervised learning classifier that we used to run the data with. Naive Bayes classifier is a family of probabilistic classifier, based on Bayes theorem. Naive Bayes assumes that features are independent from one another.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\, P(x_1, .., x_n \mid y)}{P(x_1, \ldots, x_n)}$$

with n is the number of features. NB technique uses probability to classify data based on the given probability of an occurred event.

In this study, we used MultinomialNB function from sklearn. MultinomialNB is suitable for discrete data (non-continuous data), which is our data. NB classifier has a feature called Laplace smoothing where it increases the zero probability to the small positive value to avoid discarding features (that is supposed to have zero probability) in our data. Therefore, in our MultinomialNB function, we set alpha (Laplace smoothing) parameter to be 1.0

F. Evaluation

To evaluate the performance of the algorithms we apply three methods

a) Cross-Validation for F1 Score

When it comes to measuring how accurate our model do, there are plenty of scoring methods, i.e. Accuracy, Precision, Recall, F1. We want to avoid Accuracy due to the large contribution of a class on True Positive. Our data does not have a balanced number between classes.

Precision is the probability of true Positive out of the total predicted Positive (true and false).

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

Recall captures the probability of true Positive out of our predicted values.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

F1 is used to seek a balance between Precision and Recall. Therefore, we decided to use F1 score measurement.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Instead of regularly splitting the data into 80-20 for training and test and running the data once, we use Cross-validation for the performance measurement. Cross-Validation will run the data in X iterations (in this case we chose 10 iterations) in which each iteration has a different set of training and testing data.

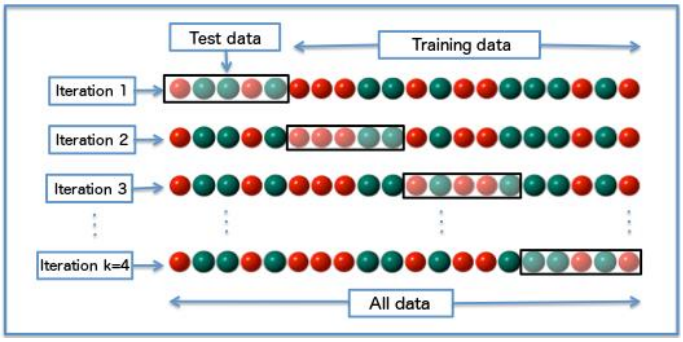


Figure 4. Cross Validation Scheme. This figure is adapted from User:Joan.domenech91 originally released under CC BY-SA 3.0

b) Learning Curve

The learning curve is used to compare the performance of training and testing data over various numbers of training instances. It gives the data of how well our model can generalize to new data [16]. The purpose of the learning curve is to minimize bias and variance and find the right number of features for a corresponding model. Often, model with more data does better.

Figure 5. Sample Learning Curve Graphs. Example is taken from our results

The learning curve also tells us when the model has learned as much as it can about the data. This occurs when:

1) The performance on training and testing sets stay stable
2) The gap between error rate stays consistent despite the increasing number of training instances

There are three types of learning curves:

1) High variance: When training and testing errors converge and are high (Figure 6)
2) High bias: When there is a large gap between the errors (Figure 6)
3) Ideal learning curve: Testing and training learning curves converge at similar values and a smaller gap between errors

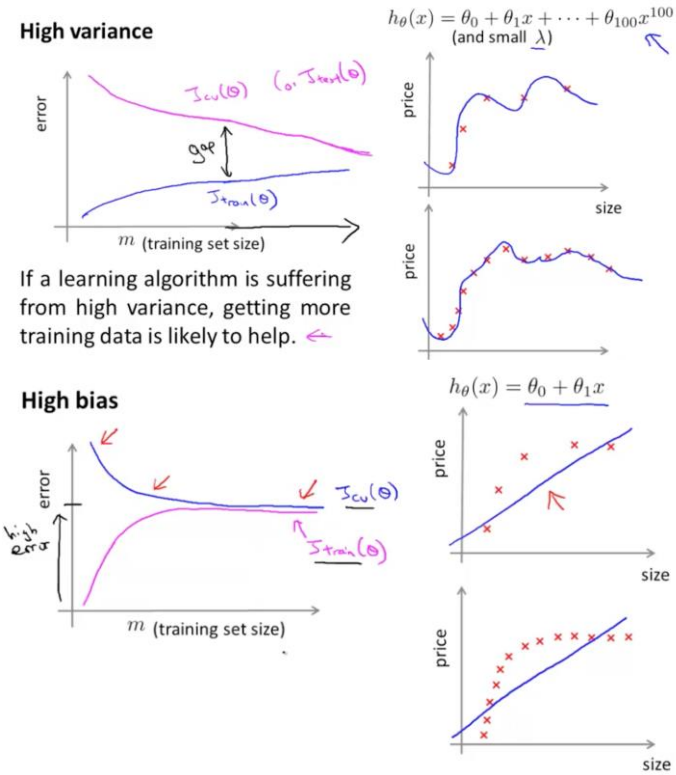


Figure 6. High Variance Learning Curve (upper figure) and High Bias Learning Curve (lower figure) Example. This figure is adapted from [15]

IV. RESULTS

In this section, we showcase three performance metrics: feature importance, f1 score and learning curve. We first calculated the importance of each feature from sklearn. Then we computed f1 score with cross-validation for three algorithms. Finally, we plotted the learning curve over different number of instances of three methods.

A. Feature Importance

We ran ExtraTreesClassifier function in sklearn to check the importance of each feature (Top 10 and Bottom 10 features are displayed in Table 1 and Table 2).

From the observed results, we chose the top six most important features are: customer's Color, Personality, Interest, Transportation, Budget, and Ambiance; bottom six least important features are: customer's Cuisine, Religion, Activity, restaurant's Hours, Days, and Name.

The results were surprising to us since we predicted that customer's features like (favorite) Color or Personality would not affect their liking for the restaurants, but the results showed that those features have the highest correlation. The least importance feature is the cuisine type that the customer likes - 0.0. What user's favorite cuisine is does not seem to affect their liking on the restaurants. The second least importance feature is the restaurant hours.

Table 1. Top 10 Features. The feature importance is computed by ExtraTreeClassifier built-in function from sklearn. The features are ranked for the partition of the dataset.

Ranking	Feature	Importance
1	Ccolor=purple	0.054304
2	Cpersonality=hunter-ostentatious	0.052972
3	Cinterest=variety	0.050607
4	Ctransport=on foot	0.044125
5	Cbudget=low	0.039327
6	Cambience=family	0.033144
7	Cbudget=medium	0.030156
8	Cdrink_level=casual drinker'	0.029640
9	Cdress_preference=informal	0.028742
10	Cpersonality=thrifty-protector	0.028575

Table 2. Bottom 10 Features. The feature importance is computed by ExtraTreeClassifier built-in function from sklearn. The features are ranked for the partition of the dataset.

Ranking	Feature	Importance
276	Ccuisine=Tunisian	0.000000
275	Ccuisine=Organic-Healthy	0.000000
274	Ccuisine=Russian-Ukrainian	0.000000
273	Ccuisine=Southwestern	0.000000
272	Ccuisine=Indonesian	0.000000
271	Ccuisine=Hungarian	0.000000
270	Ccuisine=French	0.000000
269	Ccuisine=Armenian	0.000000
268	Ccuisine=Dutch-Belgian	0.000000
267	Ccuisine=Brazilian	0.000000

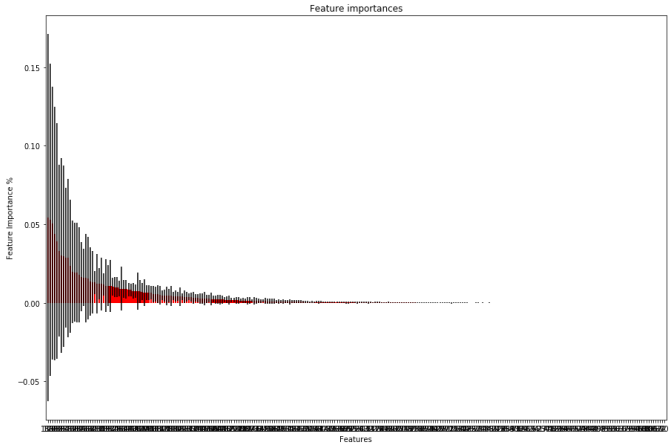


Figure 7. Importance of Features. The feature importance is computed by ExtraTreeClassifier built-in function from sklearn. The total importance percentages sums up to 1.0. The features are ranked for the partition of the dataset.

B. F1 Score

In this section, we display the results of the mean and standard deviation (SD) of f1 score for three models.

Our data is divided into 80-20, 80% of the data is for training and 20% is for testing. We used Cross-validation (CV) model to run our data for ten iterations. In each iteration, a different 80% of the data is trained and a different 20% is tested. F1 score is computed after every iteration and averaged out of ten iterations. SD is a measurement of uncertainty. It shows how far away data spreads out from the average. the higher the SD, the worse the model is doing since our f1 score since f1 varies heavily.

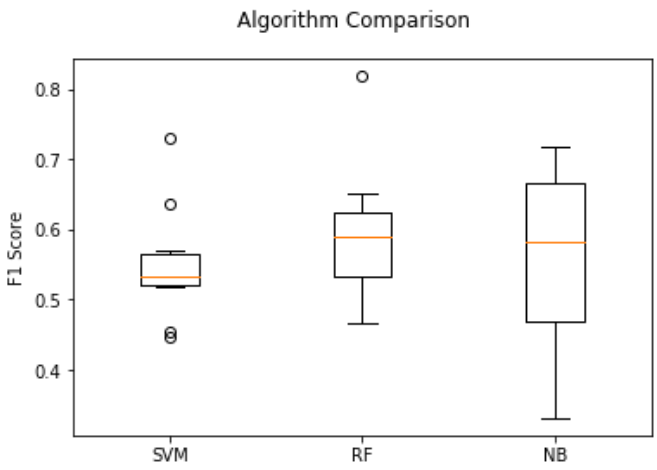
Table 3. Mean and Standard Deviation of F1 of SVM, RF, NB models. We computed Mean and Standard Deviation of our models across different variations of our dataset. Each variation contains different set of features based on their importance ranking.

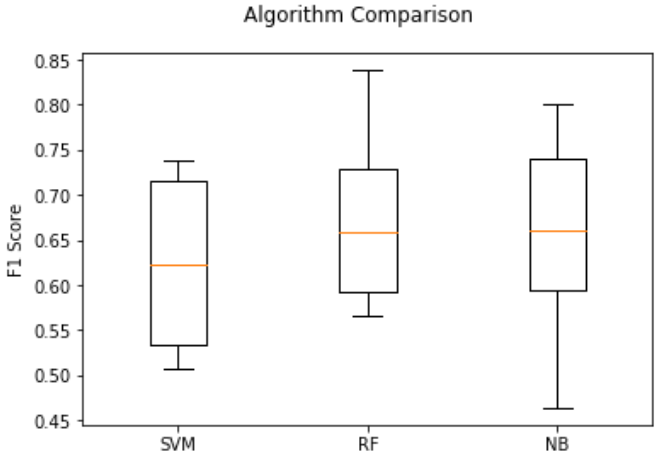
	SVM	**RF**	**NB**
Variation 1 (Least features)	0.549482 (0.0788)	0.592319 (0.0945)	0.561224 (0.1259)
Variation 2 (More important features added)	0.623701 (0.0913)	0.677312 (0.0965)	0.656376 (0.1082)
Variation 3 (Less important features added)	0.540402 (0.1119)	0.632346 (0.1077)	0.562260 (0.1121)
Variation 4 (Full features)	0.624878 (0.1112)	0.704369 (0.0851)	0.675336 (0.0939)

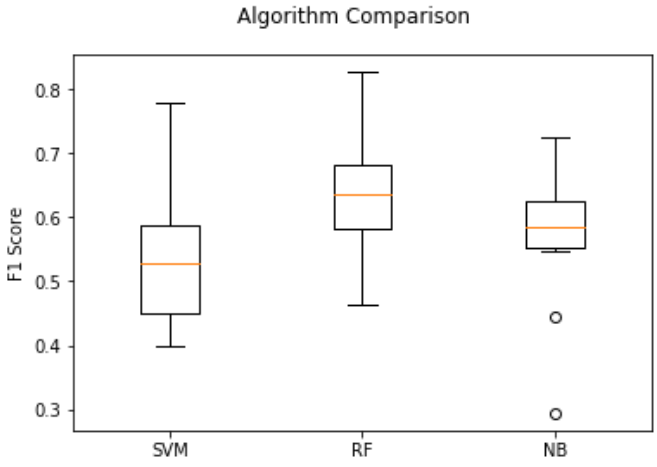
From Table 3, we can notice that our average f1 scores are low across our models (mostly below 0.7) and its SD is overly high.

Among three models, RF model results in the highest f1 score and SVM results in the lowest f1 score throughout different variations of our dataset.

Our goal was also testing on the effect of different features on our model. As the result, adding more important features increases by f1 by 10%. However, adding less important features does not improve the performance.







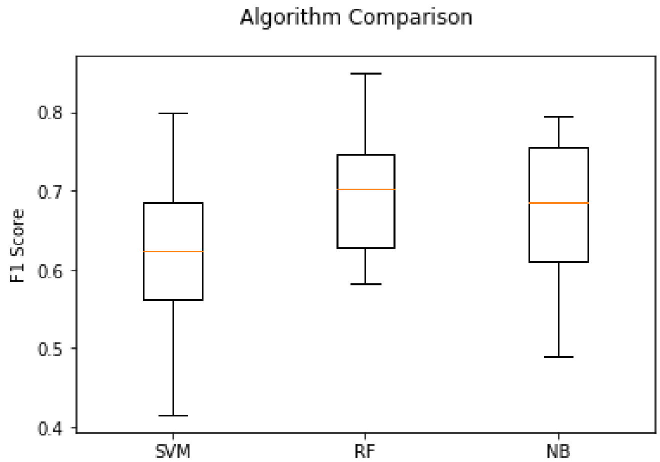


Figure 8. Boxplots of F1 Score of SVM, RF, NB Models. The uppermost boxplot represents dataset with the fewest features. The second boxplot figure has more important features added. The third boxplot figure has less important figures added. The lowermost boxplot figure represents data with full features.

C. Learning Curve

In this section, we present the learning curves of different models with different training instances. Similar to f1 process in section B, we also used cross-validation (CV) with 80-20 data and f1 score to compute the learning curve. The only difference is that in this process, the function uses different set of training samples. Training data is incremented by 1000. Since we split our data 80-20 for training and testing, the training instances contain the maximum of 8,000 samples (out of 10,000 samples). Our CV score represents the testing f1 score.

Some common characteristics of all the resulted learning curves:

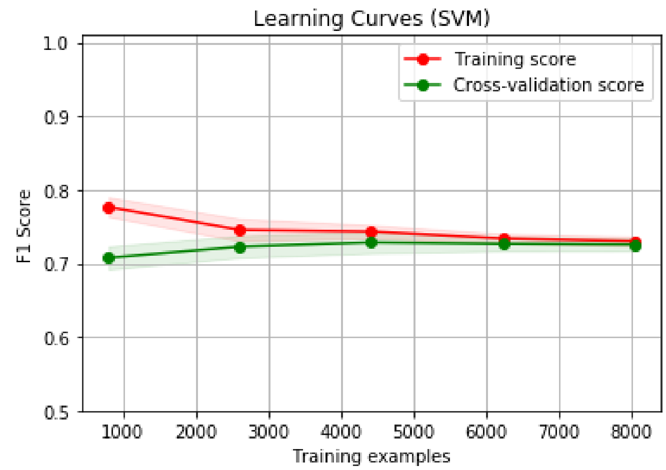
1. Adding features that have great importance rank significantly increases the performance
2. Adding features that have low important rank does not make a difference but creates higher variance on our models. This is shown by a bigger gap between training score and cv score curve; the bigger the gap, the higher the variance.

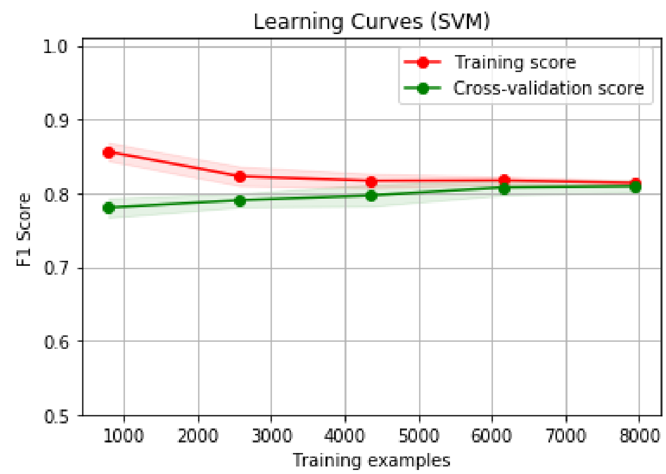
Out of three models, RF yields the highest score and NB has the lowest performance. We will go into a detailed analysis of each model and explain why having a maximum score is not a good result.

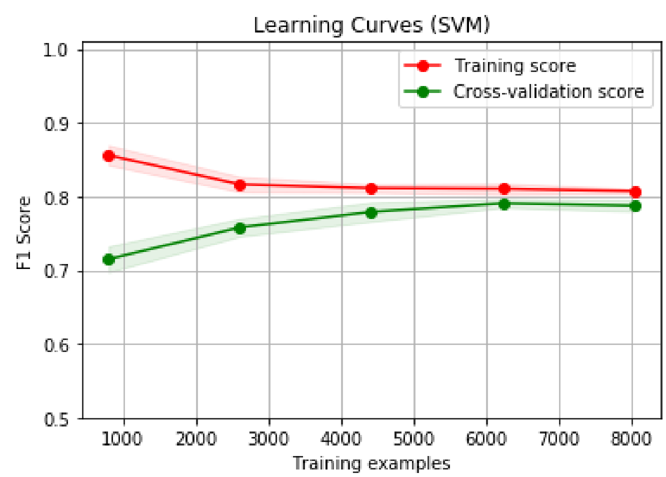
1) Support Vector Machine (SVM)

A general performance of the learning curves on four variations is the decrease of the training score line (red line) as the training examples increase. This means that our model does not have a great ability to generalize with new data. This indicates a high bias (underfitting) in our SVM model. A way to increase the performance of the underfitting model is not adding more data points but adding more features.

As we added more features, out f1 score increases by 10%. However, that does not eliminate high bias out of our model. We also noticed that as we added more irrelevant data, the training curve and CV curve don't converge anymore, which imply a higher variance in our model.







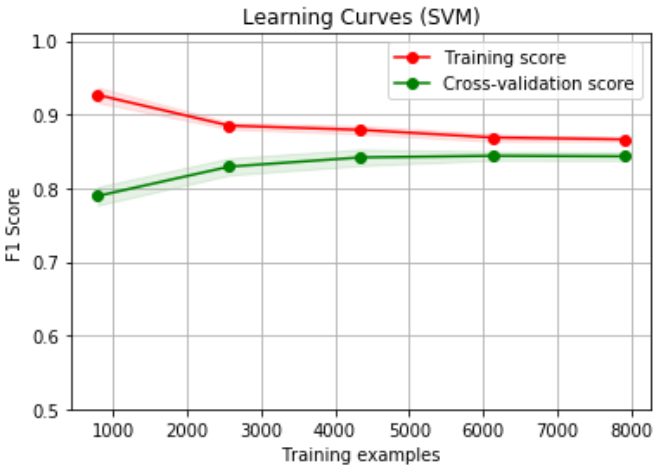


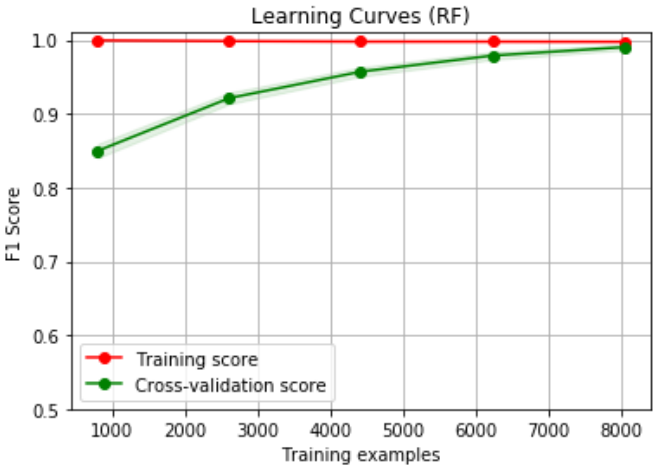
Figure 9. Learning Curves of Support Vector Machine Model. We performed CV with f1 score for SVM model with incrementing set training examples by 1000. The uppermost curves represent dataset with the fewest features. The second curves have more important features added. The third curves have less important figures added. The lowermost curves represent data with full features.

2) Random Forest (RF)

A general performance of our RF model is the maximum of the training score regardless of training examples. Maximum score means absolutely no error. However we know it is not true in this case. Thus, this represents a high variance (overfitting) in our RF model. High variance is caused by complexed model. A solution is to reduce the model complexity or add more data points.

It's evident in this case that adding more features will not help the performance. Therefore, it is easy to see there is no difference between the f1 score of the first figure (fewest features) and f1 score of the last figure (most features).

We also noticed when we added features that are less important in our model, the gap between training score and cross-validation score increased. As we mentioned above, the bigger the gap, the greater the variance. Adding irrelevant features worsened our model.



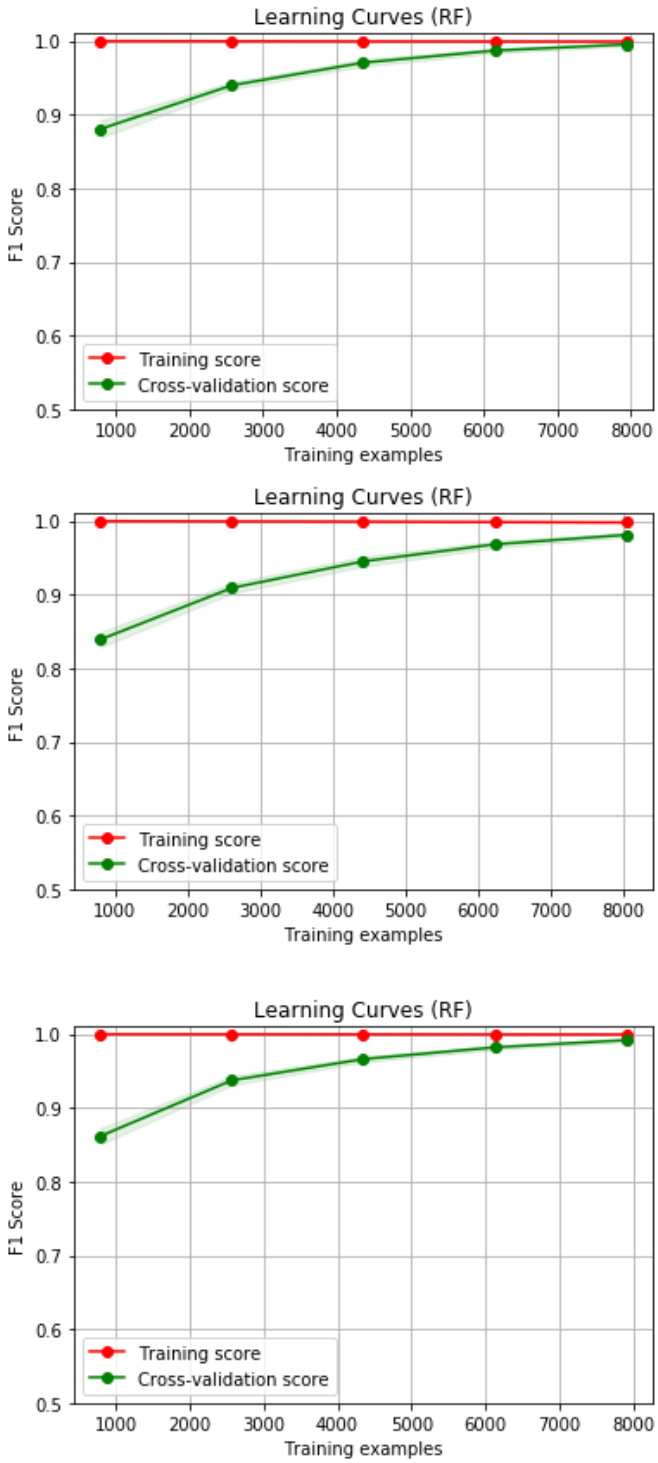
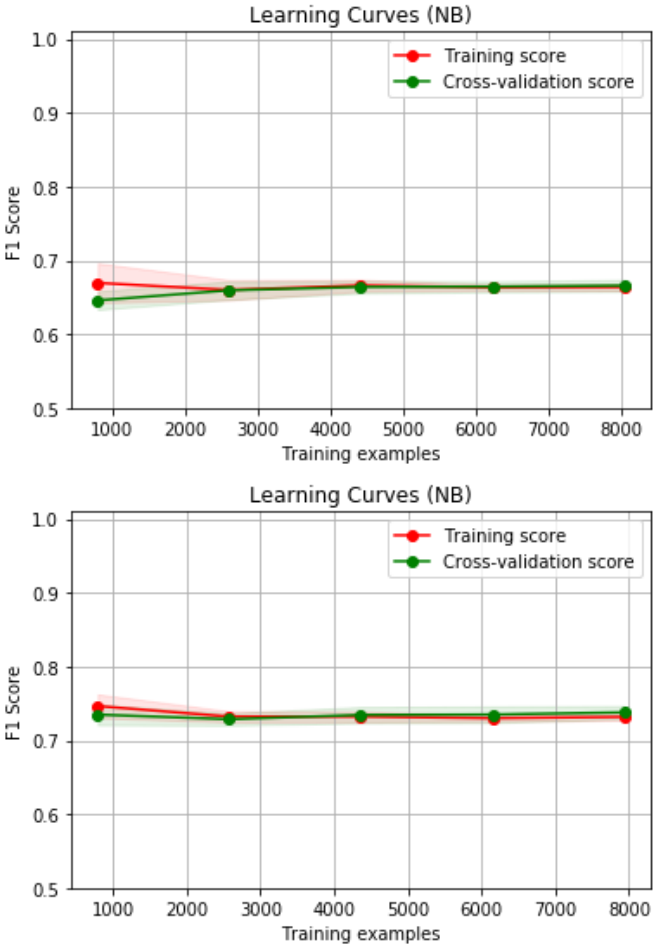


Figure 10. Learning Curves of Random Forest Model. We performed CV with f1 score for RF model with incrementing set training examples by 1000. The uppermost curves represent dataset with the fewest features. The second curves have more important features added. The third curves have less important figures added. The lowermost curves represent data with full features.

3) Naive Bayes (NB)

In this NB model, our f1 scores are the lowest compared to those of SVM and of FR. To mitigate the problem, we added more features. Similar to SVM and RF models, f1 score increased when we added more important features to the model (second figure), but when we added less important features (third, fourth figure), the gap between training score and CV score expanded. F1 score in the third figure is not as high as f1 score of the second figure.

On the good side, our NB model does not show severe underfitting or overfitting problem. The gap between training score and CV score is minimal. In the first two figures, the majority of the red line and green line are overlapping. This show there is no difference between our predicted results and the actual results.



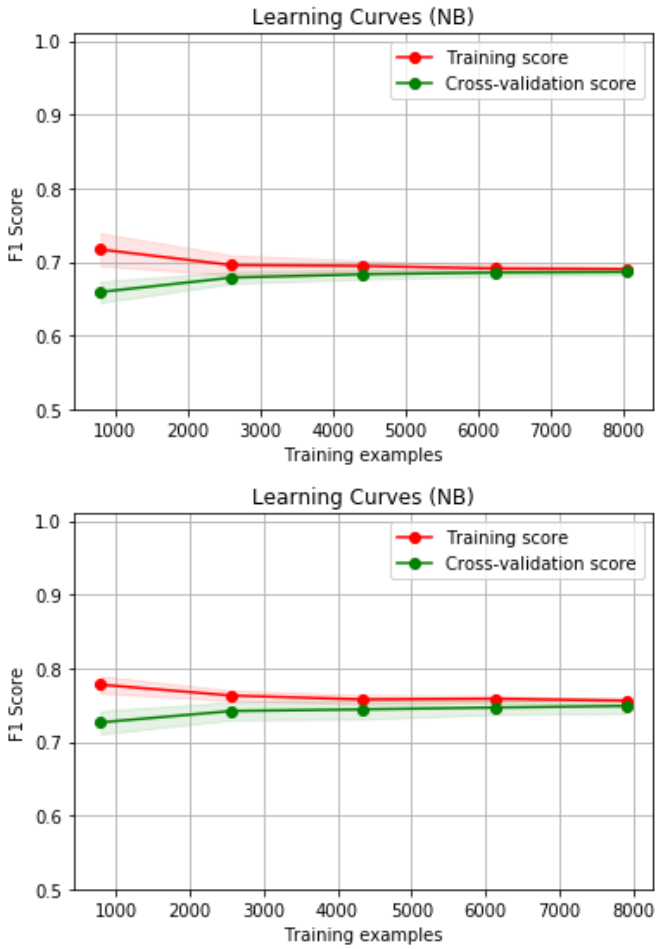


Figure 11. Learning Curves of Naive Bayes model. We performed CV with f1 score for NB model with incrementing set training examples by 1000. The uppermost curves represent dataset with the fewest features. The second curves have more important features added. The third curves have less important figures added. The lowermost curves represent data with full features.

V. CONCLUSION

In this paper, we tested three supervised learning algorithms for recommendation system for Siri: Support Vector Machine (SVM), Random Forest (RF) and Naive Bayes (NB). Although the results showed that RF has the best performance with the highest f1 score, the learning curve result proved that the model is high variance. NB models resulted in the lowest f1 scores throughout the tests.

In this approach, we were able to compare and contrast different machine learning algorithms to see which one performs the best for our type of data. In addition, our approach let us train not only one specific portion of our data but multiple random sections for higher accuracy. We were able to identify what features are likely to have a major impact on user's rating. The higher ranked features are all user's personal characteristics and the lower ranked features are restaurant's features. And we also came to the conclusion that

adding more features does not fix overfitting problem, only adding the "right" features do.

Nevertheless, we believe we could have improved our evaluation method by using of accuracy score and AUC (Area under the ROC Curve) besides f1 score. We want to manually select the values for the parameters our models rather than setting them as "auto" like right now, i.e gamma and C parameter in SVM model. A limitation we faced was the number of data points in our dataset. For future work, we want to develop a reinforcement learning algorithm and gather more data, especially browsing data, to train Siri.

VI. REFERENCES

[1] "Virtual assistant," Wikipedia, 02-Dec-2018. [Online]. Available: https://en.wikipedia.org/wiki/Virtual_assistant.

[2] T. Simonite, "'Siri, Why Have You Fallen Behind Other Digital Assistants?'," Wired, 06-Oct-2017. [Online]. Available: https://www.wired.com/story/siri-why-have-you-fallen-behind-other-digital-assistants/.

[3] S. Krishna, "Amazon is reportedly designing AI chips to improve Alexa," Engadget, 12-Feb-2018. [Online]. Available: https://www.engadget.com/2018/02/12/amazon-ai-chip-alexa-report/.

[4] https://www.quora.com/Is-Siri-an-example-of-artificial-intelligence

[5] G. Campagna and R. Ramesh, "Deep Almond: A Deep Learning-based Virtual Assistant," rep.

[6] NBC. News, YouTube, 28-Jun-2017. [Online]. Available: https://www.youtube.com/watch?v=uE_WJTnqUwA.

[7] Z. Yang and X. Su, "Customer Behavior Clustering Using SVM," Physics Procedia, vol. 33, pp. 1489–1496, 2012.

[8] RSNA Pneumonia Detection Challenge | Kaggle. [Online]. Available: https://www.kaggle.com/osbornep/reinforcement-learning-for-meal-planning-in-python/notebook.

[9] X. Wang, F. Luo, Y. Qian, and G. Ranzi, "A Personalized Electronic Movie Recommendation System Based on Support Vector Machine and Improved Particle Swarm Optimization," Plos One, vol. 11, no. 11, 2016.

[10] "KDnuggets," KDnuggets Analytics Big Data Data Mining and Data Science. [Online]. Available: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html.

[11] A. Bahnsen, "Machine Learning Algorithms: Introduction to Random Forests," DATAVERSITY, 05-Dec-2017. [Online]. Available: http://www.dataversity.net/machine-learning-algorithms-introduction-random-forests/.

[12] L. Wilson, "DJ RANDOM FOREST: SONG RECOMMENDATIONS THROUGH MACHINE LEARNING," MSiA Student Research, 09-Feb-2018. [Online]. Available: http://sites.northwestern.edu/msia/2018/02/09/dj-random-forest-song-recommendations-through-machine-learning/.

[13] "Machine Learning Algorithms Pros and Cons," HackingNote. [Online]. Available: https://www.hackingnote.com/en/machine-learning/algorithms-pros-and-cons.

[14] "Restaurant Data with Consumer Ratings," Kaggle UCI, 27-Sep-2017. [Online]. Available: https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings/.

[15] "Learning Curve," ritchieng.github.io. [Online]. Available: https://www.ritchieng.com/machinelearning-learning-curve/.

[16] N. Donges, "The Random Forest Algorithm – Towards Data Science," Towards Data Science, 22-Feb-2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[17] K. Miyahara and M. J. Pazzani, "Collaborative Filtering with the Simple Bayesian Classifier," PRICAI 2000 Topics in Artificial Intelligence Lecture Notes in Computer Science, pp. 679–689, 2000.

[18] S. Ray, "6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python)," Analytics Vidhya, 11-Apr-2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[19] N. Salerni, "Report: 98% of iPhone Users Have Tried Siri, Most Don't Use it Regularly," iPhone in Canada Blog, 06-Jun-2016. [Online]. Available: https://www.iphoneincanada.ca/news/report-98-of-iphone-users-have-tried-siri-most-dont-use-it-regularly/.

[20] S. Perez, "Siri usage and engagement dropped since last year, as Alexa and Cortana grew," TechCrunch, 11-Jul-2017. [Online]. Available: https://techcrunch.com/2017/07/11/siri-usage-and-engagement-dropped-since-last-year-as-alexa-and-cortana-grew/.

[21] Zhang, Heng-Ru & Min, Fan & He, Xu. (2014). Aggregated Recommendation through Random Forests. TheScientificWorldJournal. 2014. 649596. 10.1155/2014/649596.