Classification Performance of KNN, ANN and SVM -How Data Dimension and Size Matter

Yan Cheng
ycheng456@gatech.edu

I. INTRODUCTION

We want to know how efficient are K-Nearest Neighbor, Artificial Neural Network and Support Vector Machine algorithms on solving classification problems. We are going to build KNN, ANN and SVM models to predict whether a customer will churn and whether a patient has diabetes. We will compare the performance of the three algorithms on the same dataset. We will also compare the performance of each algorithm on two different datasets.

II. DATA AND EDA

There are three reasons why we chose the churn data and the diabetes data. First, they are high dimensional, so we can evaluate the performance of different algorithms on high dimensional data. Second, they have different size so we can evaluate the hypothesis that the model performance will improve as the data size increases. Third, there are 15 continuous features in the churn data and 18 categorical features (14 of them are binary) in the diabetes data. This allows us to evaluate how the distribution of features affect model performance.

A. Dataset 1: Churn Data Description

Dataset 1 is a telecommunication churn dataset with 20 features and 4250 instances originally. There are 18 features and 1196 instances left after we preprocessed the data. Feature customer_id is not indicative, so we decided to delete it.Feature state might has some interesting information in it. For example, there might be more market competition is some states. We may use one-hot encoding to convert the categorical values into numerical numbers. However, there are 50 different states. The dimension of features will be increased significantly if we use one-hot encoding here. Given the limited time, we chose to delete categorical feature - state. So there are 18 features left in the churn data. Some binary features are coded as "True" and "False", and they are converted to integer data type. Feature area_code has three categorical values and they are recoded as 0, 1 and 2. There are 15 continuous features. Figure 1 below shows the distribution of the features.

This dataset is very clean and there are no missing values. The features have very different scales and this dataset has imbalanced classes. For example, the maximum total_night_minutes value is 395 and the maximum value of the binary features is 1. This will cause troubles if the machine learning algorithm is sensitive to scales. There are two classes in the target labels: 1 and 0. Table I bellow shows the distribution of the taget labels is imbalanced.

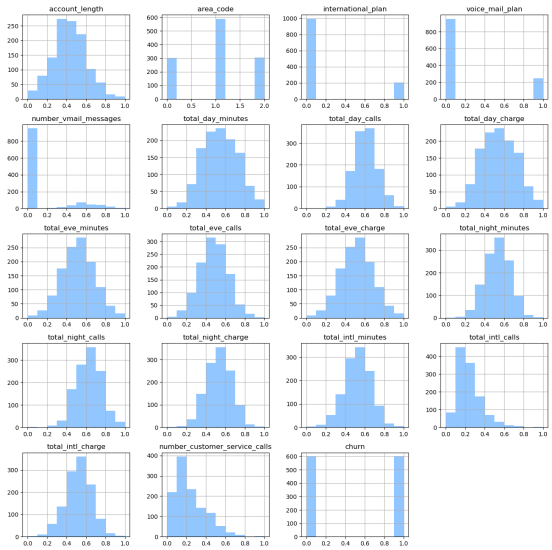

Fig. 1. Distribution of Features (Churn Data)

TABLE I
ORIGINAL TARGET LABEL DISTRIBUTION

Data	Class	Value Counts	Percentage
Churn	1	598	14.07%
	0	3652	85.93%
Diabetes	1	35097	15.29%
	0	194377	84.71%

B. Dataset 2: Diabetes Data Description

Dataset 2 is diabetes data with 21 features and 253680 instances originally. 18 features are categorical and 14 of them are binary. Age is represented by 13 categories and education is represented by 6 categories. Income is represented by 8 categories. Figure 2 below shows the distribution of features in the diabetes data.

The features are on different scales. The maximum BMI value is 98, but the maximum value of all the binary features is only 1. Like the churn data, this will cause trouble if the machine learning algorithm is sensitive to scales. Overall, this dataset is very clean. All the categorical features are encoded as numerical numbers, and there are no missing

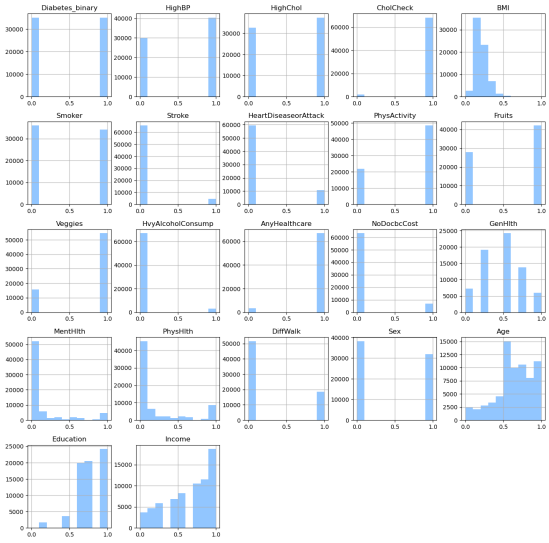

Fig. 2. Distribution of Features (Diabetes Data)

values. There are two classes in the target labels: 1 and 0. The label distribution is imbalanced too. This is shown in Table 1.

C. Data Pre-processing

We need to scale the features and handle the imbalance issue before we fit any models with the two datasets.

- **Handling Imbalance** When data is imbalanced, model performance metrics, like accuracy, can be misleading. Some models don't work very well at identifying the minority classes. For the diabetes data, if we simply predict 0 for all the test cases, the accuracy rate is going to be 84.71%, but the model failed identifying any patient that has diabetes. However, the health care providers need a model that can identify patients who have diabetes. Therefore, we need to find a way to overcome the issue with imbalanced labels. We count the number of data points in the minority class and sample the same number of data points from the majority class. We resampled both the churn and diabetes datasets.

TABLE II
TARGET LABEL DISTRIBUTION AFTER RESAMPLING

Data	Class	Value Counts	Percentage
Churn	1	598	50%
	0	598	50%
Diabetes	1	35097	50%
	0	35097	50%

- **Feature Scaling** The features in both datasets have very different scales. We used MinMaxScalar to scale the continuous features in the two datasets. The feature values fall between 0 and 1 after being scaled.

III. ALGORITHMS

KNN, ANN and SVM models are fitted to the churn data and diabetes data. Recall is used to evaluate the the model performance. For the churn data, companies really want to predict customers who will churn. The goal for the companies is to reduce churn rate and increase revenue. For the diabetes data, health care providers are interested in identifying patients with diabetes correctly, so they can provide treatment and other medical services to those patients who need it.Therefore, we believe recall will give us more accurate information about how the models do on the classification tasks at hand.

- **Algorithm Hypothesis** From the exploratory data analysis we did, we hypothesize that KNN models will not perform well on these two datasets due to high dimension of the data. All three models will perform better on the diabetes problem because there are more data samples.
- **Data Splitting** Before we fit the models, we split the data into a training set and a testing set.

TABLE III
DATA SPLITTING

Data		Size	Percentage
Churn	Training Set	837	70%
	Testing Set	359	30%
Diabetes	Training Set	49136	70%
	Testing Set	21058	30%

A. K Nearest Neighbors

KNN classifies a new sample using the K-closest samples from the training set. This method depends on how the distance between samples is defined. It operates on the principle that similar instances are close to each other in the feature space. Given a new instance, KNN identifies the 'k' closest training instances (neighbors) and classifies the new instance based on the majority class of its neighbors. The algorithm first calculates the distance between the new instance and all training instances, second selects the k-nearest neighbors based on the calculated distances, and last assigns the class label that is most frequent among the k-nearest neighbors.

- **Distance Metric and the Number of Neighbors** KNN depends on how the distance between samples is defined, so we explored several distance metrics when we fit the KNN model to the two datasets.

TABLE IV
KNN WITH DIFFERENT DISTANCE METRICS

Data		Euclidean	Manhattan	Cosine	L2
Churn	K	5	7	5	5
	Recall	0.7654	0.7765	0.7654	0.7654
Diabetes	K	15	15	15	15
	Recall	0.7520	0.7525	0.7613	0.7518

The performance of KNN with Manhattan distance is slightly better than other distance metrics on churn data, and the performance of KNN with Cosine distance is slightly better than other distance metrics

on diabetes data. This is shown in Table IV. However, the learning curves have weird behavior if we use cosine distance metric on the diabetes data. So we used cross validation to choose the value of K with Manhattan distance metric on both datasets.

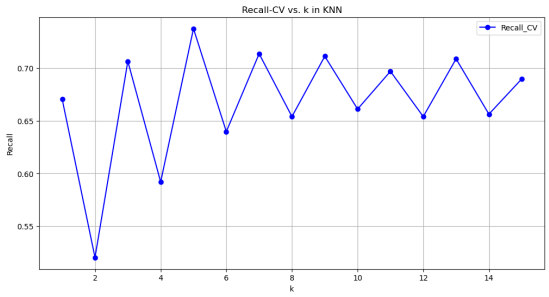

Fig. 3. CV Recall VS K in KNN (Churn Data)

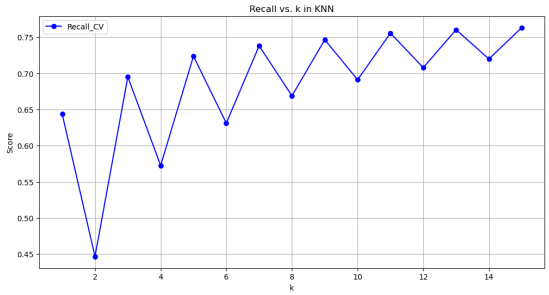

Fig. 4. CV Recall VS K in KNN (Diabetes Data)

From Figure 3 and 4, we can see that the KNN performance is very unstable on both datasets.

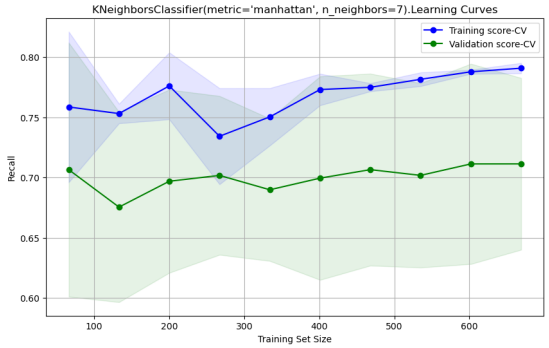

Fig. 5. KNN Learning Curves (Churn Data)

From the learning curves in Figure 5 and 6, we can see that there is a big gap between training recall and testing recall. It means that the KNN models have high variance on the churn dataset and diabetes dataset. Meanwhile, the training recall and validation recall are smaller than 0.8 on both datasets. This is to say the KNN models have high bias and high variance on both datasets. This is because as the number of dimensions increases, the size of the data

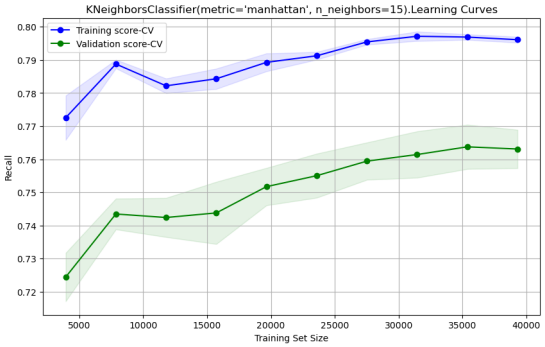

Fig. 6. KNN Learning Curves (Diabetes Data)

that we need to improve the model performance grow exponentially. If we take a look at the shaded green area in Figure 5 and 6, the variance of the validation recall is much higher on the churn data than that of the diabetes data. This is because the churn data is much smaller than the diabetes data.

As KNN is a distance-based method, it gets directly impacted by high dimensional data. As individual distances between the points get less significant in high dimensional spaces, the dimensional space becomes uniform. As the number of dimensions increases, the volume of the space increases exponentially, making the data sparse. This sparsity makes it difficult for KNN to find meaningful neighbors, as the distances between points become more uniform. This distance uniformity makes it difficult to distinguish between close and far points and reduces the effectiveness of distance-based algorithms like KNN. This also leads to poor generalization. This is why KNN models don't perform very well on both datasets because both diabetes data and churn data are high dimensional.

B. Neural Network

– **Architecture of the Neural Network**

It's very important to configure the architecture of a Neural Network model. How many layers do we need? How many neurons do we need in each layer? What activation function we are going to use in each layer? We will discuss all these questions one by one.

∗ **Depth of the Neural Network**

How many layers do we need? The more layers an ANN has, the deeper it is. A deep Neural Network usually has more than 2 hidden layers. The deeper a Neural Network is, the more complicated and flexible it is. Deep Neural Networks can usually handle more challenging problems. However, for simple problems, deep Neural Networks tend to overfit.

For the two binary classification problems we have for this assignment, shallow networks perform

better than deep networks. Therefore, the Neural Networks we built have four layers in total: input layer, two hidden layers, and output layer.

* **Input and Output Layer**
  The size of input and output layer is determined by the data and the problem we are going to solve. In Neural Network, each data point will be passed to the input layer first, and then the input layer will pass the data to the first hidden layer. So the input layer has the same shape as the input data. This is done by code: Input(shape=(X_train.shape[1],)). For the churn data, we have 18 neurons in the input layer since we have 18 features. For the diabetes data, we have 21 neurons in the input layer because there are 21 features. We don't need activation function for input layer.
  We only need one neuron in the output layer for binary classification problems. We choose sigmoid as activation function, so the network will output one number which is a value between 0 and 1.
* **Width of the Neural Network**
  As we discussed above, the Neural Networks we built for the two classification problems have two hidden layers. How many neurons do we need in each hidden layer? In a neural network, the number of neurons in the hidden layer corresponds to the complexity of the model generated to map the inputs to output (s). More neurons create a more complex function (and thus the ability to model more nuanced decision boundaries) than a hidden layer with less nodes.
  We used cross validation to find good values of neuron numbers in the two hidden layers. In general, networks with wider 1st hidden layer perform better than the ones with a narrower 1st hidden layer; networks with narrow 2nd hidden layer perform better than the ones with a wider 2nd hidden layer. We used ReLU as the activation function for the two hidden layers.

– **Learning Rate and Efficiency of the Neural Network**
  The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.
  From figure 7, we can tell that learning rate = 0.008 for a network with the shape of (60,5,1) is the best on the churn data. However, if we set learning rate = 0.01 for a network with the shape of (60,5,1), it actually performs better.

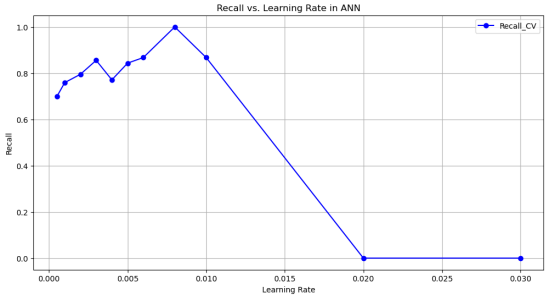

Fig. 7. CV Recall VS Learning Rate in ANN (Churn Data)

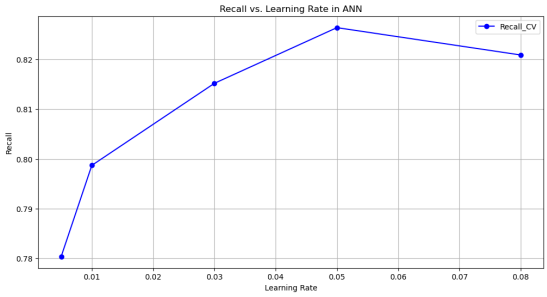

Fig. 8. CV Recall VS Learning Rate in ANN (Diabetes Data)

From figure 8, we can tell that learning rate = 0.05 for a network with the shape of (100,20,1) performs better on the diabetes data.

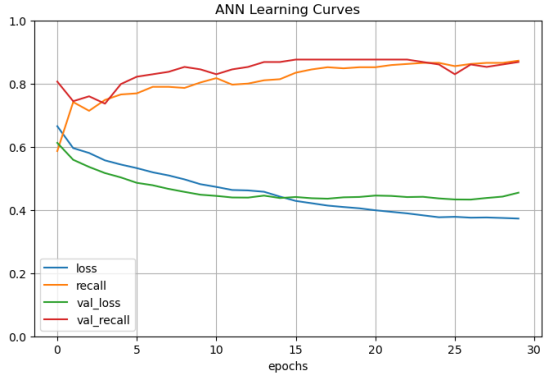

Fig. 9. ANN Learning Curve (Churn Data)

From figure 9, we can see that the ANN learns very well from the churn data. As the model iterates more, the recall from training set and validation set keep increasing and losses keep decreasing.
However, the ANN model doesn't learn very well from the diabetes data. It stopped learning only after the first 10 or 20 iterations. The performance doesn't improve for the diabetes data as the model iterates more. It doesn't overfit either. This is a very interesting behavior.

– **Other Hyperparameters in the Neural Network**
  The ANN performs better on churn data with an

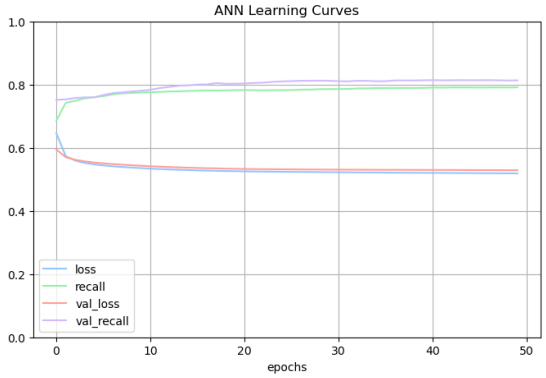

Fig. 10. ANN Learning Curve (Diabetes Data)

Adam optimizer, and it performs better with Stochastic Gradient Descent optimizer on diabetes data. ANN tends to overfit on both datasets with large values of epochs.

TABLE V
ANN MODELS FOR CHURN DATA AND DIABETES DATA

	Churn Data	**Diabetes Data**
Input Layer	18	21
1st Hidden Layer	60	100
2nd Hidden Layer	5	20
Output Layer	1	1
Learning Rate	0.01	0.05
Optimizer	Adam	SGD
Batch Size	30	200
Epochs	30	50
Training Size	837	49136
Testing Size	359	21058
Test Recall	0.88	0.80
Fit Time	2.0666	8.3743
Predict Time	0.0806	0.4615

The size of the ANN model fitted on the diabetes data is much bigger than the one fitted on the churn data which means it's more complicated, flexible, expressive and powerful. The size of the training data is also much bigger in the diabetes diagnosing problem. So the model performance should be better on the diabetes data. However, the test recall on the churn data is much higher than that of the diabetes data. This is not what we expected. A powerful Neural Network model is not learning from the data because there is not so much to learn in the data. There are 18 categorical features in the biabetes data, and 14 of them are binary. Among the 14 binary features, class distribution is imbalanced in 8 of them. The majority class is 1 in 4 features, and the majority class is 0 in another four features. There are not so many insights our model can extract from the data. No matter how powerful our model is and how long we train the model, the performance will not improve.

C. Support Vector Machine

Support Vector Machine is an extension of support vector classifier. SVC requires that the classes be separable by a linear boundary. SVM uses kernels to enlarge the feature space to address non-linearity. The most used two kinds of kernels are polynomial kernel and radial kernel. We fitted SVM models with these two kernels on the churn dataset and diabetes dataset.

– **SVM with Polynomial Kernel**
The best value of hyperparameter C we found through cross validation is 2 on both datasets. C is the cost of violations. We are more tolerant of violations when we increase the value of C, and more support vectors will be included in deciding the decision boundaries and making predictions. When C is large, the classifier fits the data less hard, so the classifier is more biased but may have low variance. When we decrease C, less support vectors will be included in deciding the decision boundaries and making predictions. The classifier fits the training data very hard which leads to overfitting, so the classifier will have low bias and high variance.
From figure 11 and 12 we can see that as we increase C the model performance increases at the beginning until C=2 when we fit the model on the churn data and the diabetes data. After that the model performance starts to decrease as we keep increasing C on both datasets which is a sign of overfitting.

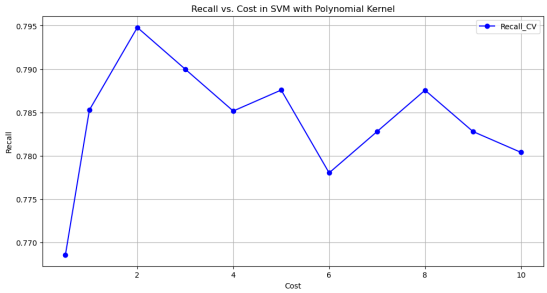

Fig. 11. CV Recall VS C in SVM-Poly (Churn Data)

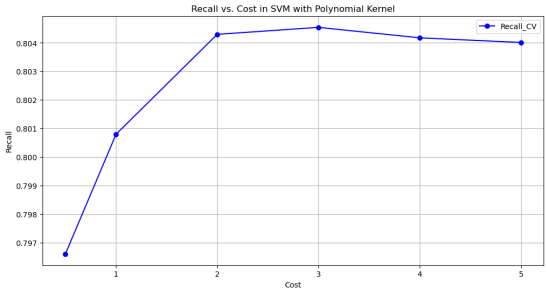

Fig. 12. CV Recall VS C in SVM-Poly (Diabetes Data)

The best value of hyperparameter d we found through cross validation is 2 on both datasets. d is the

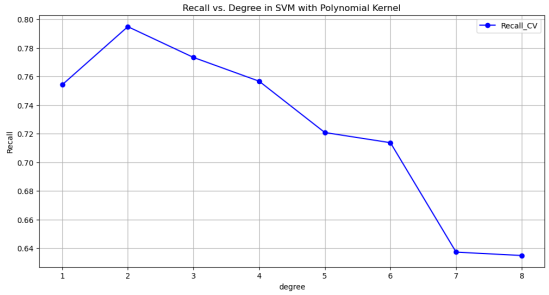

Fig. 13. CV Recall VS d in SVM-Poly (Churn Data)

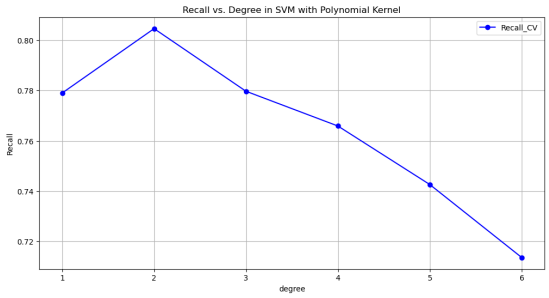

Fig. 14. CV Recall VS d in SVM-Poly (Diabetes Data)

degree of the polynomial kernel in the SVM model. As we increase the value of d, we increase model flexibility. Because a model with a higher degree has more degrees of freedom, it can take on more complicated function shapes. However, if a model is too complicated and too flexible, it tends to overfit, then it doesn't generalize very well, so it doesn't perform well on a validation set.

This is shown in Figure 13 and 14. As we increase d from 1 to 2, the model performance increases on both datasets. After that the model performance starts to decrease as we keep increasing d which is a sign of overfitting. The validation curve of d have very similar trend on the two datasets.

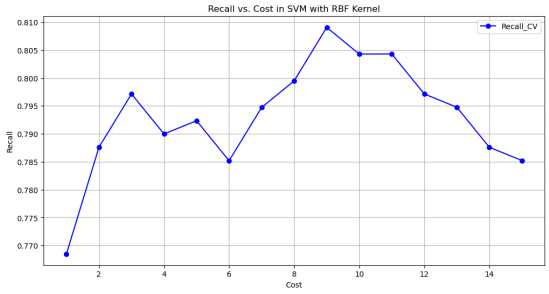

Fig. 15. CV Recall VS C in SVM-RBF (Churn Data)

– **SVM with Radial Kernel**

The best value of hyperparameter C we found through cross validation in SVM with radial kernel is 9 for the churn data and 2 for the diabetes data.

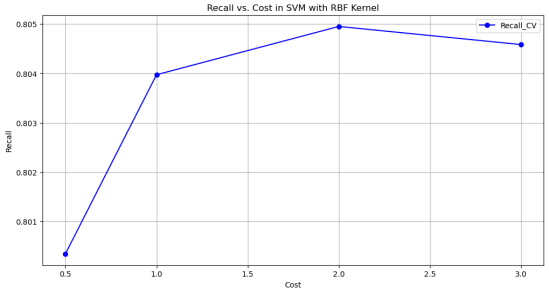

Fig. 16. CV Recall VS C in SVM-RBF (Diabetes Data)

The C parameter in SVM with radial kernel works in the same way as in the SVM with polynomial kernel. From figure 15 and 16, we can tell that the model performance increases on both datasets as we increase C at the beginning. The performance starts to decrease at C=9 as we keep increasing C on the churn data which means the model starts to overfit. On the diabetes dataset, the model achieves its best performance at C = 2.

The best gamma we found through cross validation for churn data is 0.3 and 0.05 for the diabetes data. γ is a hyperparameter in the radial kernel:

$K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$

We know a kernel is a function that quantifies the similarity of two observations. If a given test observation is far from a training observation in terms of Euclidean distance, then $\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$ will be large, and

$\exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$ will be very tiny. It means the training observations that are far from the test observation will play essentially no role in the predicted class label for the test observation. For large values of gamma, $\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$ is larger, and $\exp(-\gamma \sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$ is smaller. Therefore, if we increase gamma, we increase the flexibility of the model and decrease the number of training observations that are included in predicting class label for the test observation. The effect of increasing gamma is like decreasing K in K-Nearest Neighbor.

If gamma is too big, only the training observations that are very close or similar to the test observation will be considered when making predictions. The model will be too complicated and too flexible. It fits the training data too hard which will lead to overfitting and decrease the generalizing ability of the model. So large gamma may lead to low bias and high variance. If gamma is too small, the training observations that are not similar to the test observation will be considered when making predictions. The model will be too simple which leads to underfitting and decrease the performance of the model. So small gamma may lead to high bias and low variance.

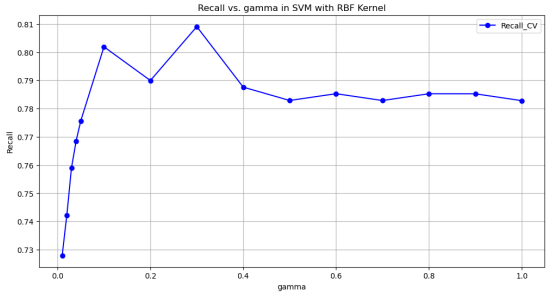

Fig. 17. CV Recall VS γ in SVM-RBF (Churn Data)

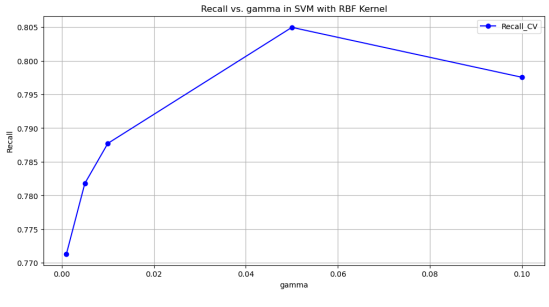

Fig. 18. CV Recall VS γ in SVM-RBF (Diabetes Data)

In Figure 17 and 18, the SVM performance increases as we increase the gamma value at the beginning, but after a certain point, 0.3 for the churn data and 0.05 for the diabetes data, the performance starts to decrease as we keep increasing gamma which means the model is becoming too flexible and it starts to overfit.

– **Comparison of SVM with Different Kernels on the Two Datasets**

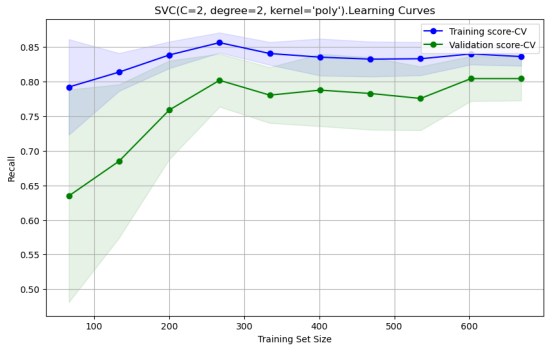

Fig. 19. SVM-Poly Learning Curve (Churn Data)

From Figure 19,20,21 and 22, we can tell all four models learn very well and fit the data very well. When the training size increases, both the training recall and the validation recall increases, and the gap between the training and validation score becomes smaller and smaller in all four models.

However, from the four figures we can see that the SVM models fitted to the churn data have higher

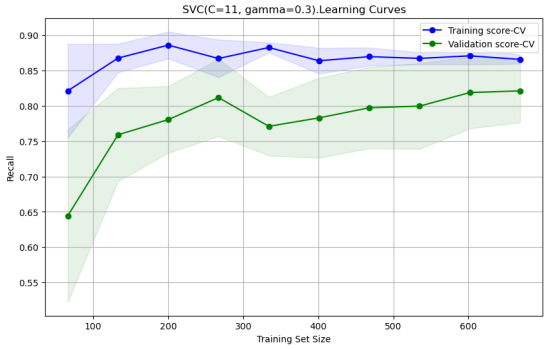

Fig. 20. SVM-RBF Learning Curve (Churn Data)

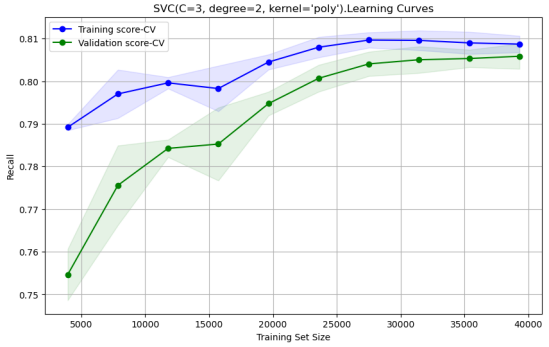

Fig. 21. SVM-Poly Learning Curve (Diabetes Data)

variance than the SVM fitted to the diabetes data. This is most likely due to the training size of the churn data is much smaller than the diabetes data.

From Table VI, we know that SVM with radial kernel perform slightly better than SVM with polynomial kernel on both datasets. SVM with radial and polynomial kernel both perform better on the churn dataset than the diabetes set. However, the fit time and predict time of the SVM models increased significantly as we increase the data size.

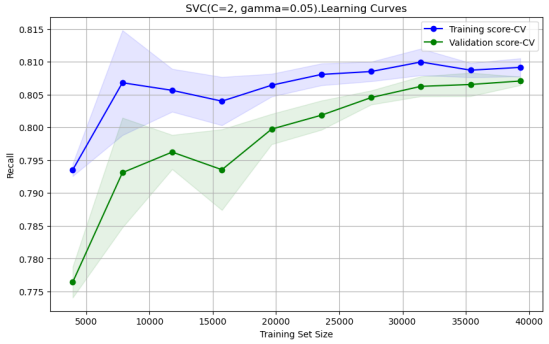

Fig. 22. SVM-RBF Learning Curve (Diabetes Data)

TABLE VI
COMPARISON OF SVM MODELS WITH DIFFERENT KERNELS

	Churn	Diabetes
Model	SVM-POLY(C=2, d=2)	SVM-POLY(C=3, d=2)
Fit Time	0.01	47.1451
Predict Time	0.0055	8.1065
Test Recall	0.8045	0.8008
Model	SVM-RBF(C=11, γ = 0.3)	SVM-RBF(C=2, γ = 0.05)
Fit Time	0.0136	47.7056
Predict Time	0.0111	32.6279
Test Recall	0.8268	0.8018

IV. COMPARISON AND DISCUSSION

A. Model Performance

From table VII below, we can see that KNN models have bad performance on both datasets due to curse of dimensionality. This proves that our hypothesis at the beginning is valid. ANN and SVM with radial kernel have the same performance on the diabetes data. ANN performs much better than SVM on the churn data.

TABLE VII
MODEL PERFORMANCE (RECALL)

Data	KNN	Neural Network	SVM-RBF
Churn	0.7765	0.88	0.83
Diabetes	0.7524	0.80	0.80

We also hypothesized that the three models would perform better on the diabetes data because of the sample size. However, all the three models have higher performance on the churn data. This is contradictory to our expectations. We discussed about the diabetes data in the Neural Network section. All three models perform better on the churn data for the same reason. There is not so much information in the diabetes data for the models to learn though it has a much bigger sample size. On the other hand, our models can certainly learn more from the churn dataset with 14 normally distributed continuous features.

B. Computation Time

TABLE VIII
FIT TIME AND PREDICT TIME FOR THE MODELS IN SECONDS

Data		KNN	Neural Network	SVM-RBF
Churn	Fit Time	0.0	2.0666	0.0038
	Predict Time	0.0156	0.0806	0.0044
Diabetes	Fit Time	0.0	8.3743	47.7056
	Predict Time	1.6108	0.4615	32.6279

From Table VIII above, we can see that the time cost of fitting KNN model is zero. This is because K Nearest Neighbor algorithm is instance-based learning. The model simply stores the training samples at fitting time. Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target label for the new instance. The Neural Network models have moderate fit time and moderate predict time. The fit time and predict time doesn't increase a lot even when the model is fitted to data with a much bigger size(the diabetes data). However, The fit time and predict time of the SVM model with radial kernel increases significantly when it's fitted to data with a much bigger size(the diabetes data).
Therefore, Artificial Neural Network models have the best performance and moderate computational cost even when the data is big. This makes ANN a very good choice if we want our machine learning model to be scalable to big data.

C. Limitations

There are some questions about our experiments left unanswered. How the different distance metrics affect KNN model? Why the ANN performs better on churn data with an Adam optimizer? Why it performs better with Stochastic Gradient Descent optimizer on diabetes data? How batch size and epochs affect the performance of a Neural Network? Why the performance and learning curves of ANN are unstable? Is there intrinsic randomness in the Neural Network algorithm? If our churn data size is bigger, will the performance of the three models improve? Given the limited space we have for this assignment. We will keep looking for answers for all these questions in our machine learning journey.

REFERENCES

[1] https://machinelearningmastery.com
[2] https://towardsdatascience.com
[3] https://www.geeksforgeeks.org
[4] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, "An Introduction to Statistical Learning", Springer, 2017.
[5] Max Kuhn and Kjell Johnson, "Applied Predictive Modeling", Springer, 2013.
[6] Aurelien Geron, "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow ", O'REILLY, 2019.
[7] Tom M. Mitchell, "Machine Learning", McGraw-Hill, 2019.