Occupations Classifier

Roger Cheng
University of California, San Diego
chc467@eng.ucsd.edu

Yoo Jin Chung
University of California, San Diego
yjc030@eng.ucsd.edu

Jay Rainton
University of California, San Diego
jrainton@ucsd.edu

Ethan Nagola
University of California, San Diego
enagola@ucsd.edu

1 Task Assignment

We all looked over the data together to find the best dataset for our project.

Roger Cheng: Researched and implemented the random forest model for this project. Wrote out the approaches and results section of the paper for the random forest model. Helped find research papers in SVM and neural networks.

Yoo Jin Chung: Processed the data and translate it into usable data. Visualized the data to decide the import features. Wrote the data preparation section under approaches. Researched and implemented SVM model for this project. Wrote out the approaches and results section of the paper for the SVM model.

Ethan Nagola: Wrote the motivation, previous works, intended contributions, organization of paper, and possible future works sections, solved the KKT conditions of the problem, helped discuss data preparations, and researched possible neural network algorithms for us to use.

Jay Rainton: Wrote out the Primal and solved the Dual formulation of the paper. Implemented and researched different neural networks to use. Wrote the neural network section for approaches and results. Wrote the conclusion. Added neural network section to possible future works.

2 Introduction

2.1 Motivation

We were really intrigued in SVM as learned in class and wanted to explore its limits. We read a few articles and came across an article titled *Comparing Techniques for Multiclass Classification Using Binary SVM Predictors* which discussed using SVM to classify multiple classes in the data. After discussing, we decided that we want to apply SVM to classify multiple classes in our data. After looking through many different data sets, we found an interesting data set with the real world application of being able to predict people's occupation based on certain factors which we planned to do with our multi-class SVM classifier. We are focusing on one-to-all SVM, as this is suggested to provide better results as well as faster run-time.[1] We also want to compare our SVM model to other commonly used methods such as Random Forest and ANN to see if we can find the best method. After looking through the data we kept the distinctive attributes to use with our classifier and thus began our journey of this project the 'Occupations Classifier'.

2.2 Previous Works

The data that we used came from University of California, Irvine's Machine Learning Repository that used the data to predict whether the salary of a row of data given the el-

ements of that row is above or below a certain threshold. In their case the threshold they set was $50,000. Some other works with this data that we looked at include *Learning and evaluating classifiers under sample selection bias* by Bianca Zadrozny [2] which tried to verify the effects of sample selection bias experimentally by applying Naive Bayes, logistic regression, C4.5 and SVMLight. This work assumed that the original dataset was not biased and artificially simulate a bias value. Another work we looked at that used this data was *Gradient Projection Methods* by Thomas Serafini et al. [3] which used both this UCI Adult data and an additional MNIST database of handwritten digits to optimise the same threshold of salary as mentioned above. Lastly, we looked at *Detecting Group Differences: Mining Contrast Sets* by Stephen D. Bay and Michael J. Pazzani [4] which used the data to provide a search algorithm for mining contrast sets with pruning rules that drastically reduce the computational complexity explicitly controlling the probability of Type I error (false positives) and guarantee a maximum error rate for the entire analysis by using Bonferroni corrections.

In addition to works with the same data set, we also looked at the article *Occupation inference through detection and classification of biographical activities* by Elena Filatova and John Prager [5] which also worked with classifying by occupation and was part of what inspired us to create an optimization problem that involved using the data set that we chose in the way that we did (predicting a data's occupation).

2.3 Intended Contributions

One intended contribution is that given information in a neighborhood, the classifier can use that information to predict occupation roles and then, based on the number of people classified in certain occupations we can take this data to companies with those same available openings to advertise in those neighborhoods allowing companies to more easily fill jobs helping both the companies and the people of the communities. We can use this same strategy to contribute to more targeted advertising as we can use the occupation of people to gain information about things they like, need, want, etc (see possible future works).

2.4 Organization of the Paper

The rest of the paper is organized as follows. In Section 3, we describe the statement of problem including its primal and dual formulations as well as its KKT conditions. In Section 4, we discuss the data preparation and the different approaches that we tried to use to solve the problem. In Section 5, we go over the results of our different approaches, share our conclusion based on those approaches, and explain possible future works that our solution can lead to. Finally, in Section 6, we provide the references to the papers we used in this research paper.

3 Statement of the Problem

Since this is a classification problem we would like to use SVM as our convex optimization solution. However, since this dataset might not be completely distinct sets we would need to have a slack variable. In addition, we would need to turn this into multiple boundary planes. This results in the primal problem:

$$\text{minimize} \sum_{j=1}^{k} ||w_j||^2 + C \sum_{i=1}^{n} \epsilon_i$$

$$\text{st. } w_{y^{(i)}} * x^{(i)} + b_{y^{(i)}} - w_k * x^{(i)} - b_k \geq 1 - \epsilon_i \; \forall \; i, \text{ and for each } k \neq y^{(i)}$$

$$\epsilon_i \geq 0 \; \forall \; i$$

3.1 Dual Formulation

So we first generate the Lagrange problem as shown below.

$$L(w, b, \epsilon, \lambda, v) = \sum_{j=1}^{k} ||w_j||^2 + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \sum_{j=1, j \neq y^i}^{k} \lambda_{ij} (1 - \epsilon_i - (w_{y^i} * x^i) - b_{y^i} + (w_j * x^i) + b_j) - \sum_{i=1}^{n} v_i \epsilon_i$$

Then we must calculate the gradient in terms of the weight and bias term. Since there are n(k-1) constraints, we need to account for when the weights and biases are are assigned to the data point and when they are not. That is why we split up the terms into two sections (S & T, R & E) so that it would be easier to read. S corresponds to when the weight term is part of the i[th] training point. This would have (k-1) for each data point, thus n(k-1) summation. T corresponds to when the weight term is not part of the i[th] training point. Thus it will have n summation for each one. R corresponds to the bias term for the i[th] training point and E corresponds to the bias term for the other k-1.

$$\bigtriangledown_{w_j} L(w, b, \epsilon, \lambda, v) = 2w_j - S_j + T_j$$

$$S_j = \sum_{i=1}^{n} \sum_{p=1, p \neq y^i}^{k} \lambda_{ip} x^i \text{ when } j = y^i$$

$$T_j = \sum_{i}^{n} \lambda_{ij} x^i \text{ when } j \neq y^i$$

$$w_j = \frac{S_j + T_j}{2}$$

$$\bigtriangledown_{b_j} L(w, b, \epsilon, \lambda, v) = E_j - R_j$$

$$R_j = \sum_{i=1}^{n} \sum_{p=1, p \neq y^i}^{k} -\lambda_{ip} \text{ when } j = y^i$$

$$E_j = \sum_{i}^{n} \lambda_{ij} \text{ when } j \neq y^i$$

$$\bigtriangledown_{\epsilon_i} L(w, b, \epsilon, \lambda, v) = C - (v_i + \sum_{j=1, j \neq y^i}^{k} \lambda_{ij})$$

with this we can formulate the dual problem as:

$$\text{maximize } \sum_{j=1}^{k} ||\frac{S_j + T_j}{2}||^2 + \sum_{i=1}^{n} \sum_{j=1, j \neq y^i}^{k} \lambda_{ij} (1 - (\frac{S_{y^i} + T_{y^i}}{2} * x^i) + (\frac{S_j + T_j}{2} * x^i))$$

$$\text{st. } C - (v_i + \sum_{j=1, j \neq y^i}^{k} \lambda_{ij}) = 0 \; \forall \; i$$

$$E_j - R_j = 0 \; \forall \; j$$

$$\lambda_{ij} \geq 0 \; \forall \; i \text{ and } \forall \; j \text{ where } j \neq y^i$$

$$v_i \geq 0 \; \forall \; i$$

3.2 KKT Conditions

given our primal problem

$$\text{minimize} \sum_{j=1}^{k} ||w_j||^2 + C \sum_{i=1}^{n} \epsilon_i$$

$$\text{st. } w_{y^{(i)}} * x^{(i)} + b_{y^{(i)}} - w_k * x^{(i)} - b_k \geq 1 - \epsilon_i \; \forall \; i, \text{ and for each } k \neq y^{(i)}$$

$$\epsilon_i \geq 0 \; \forall \; i$$

and our Lagrangian

$$L(w, b, \epsilon, \lambda, v) = f(w, b, \epsilon) + v^\top \mathbf{g}(w, b, \epsilon) + \lambda^\top \mathbf{h}(w, b, \epsilon)$$

$$L(w,b,\epsilon,\lambda,v) = \sum_{j=1}^{k} ||w_j||^2 + C\sum_{i=1}^{n}\epsilon_i - \sum_{i=1}^{n} v_i\epsilon_i + \sum_{i=1}^{n}\sum_{j=1,j\neq y^i}^{k} \lambda_{ij}(1-\epsilon_i-(w_{y^i}*x^i)-b_{y^i}+(w_j*x^i)+b_j)$$

We want to create the KKT Conditions as such

$$\nabla f(x^*) + \sum_{i=1}^{m} v_i \nabla g_i(x^*) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*) = \mathbf{0}$$

Where

$$x^* = (w, v, \epsilon)$$

The gradient of the Lagrangian as shown above is

$$\bigtriangledown_{w_j} L(w,b,\epsilon,\lambda,v) = 2w_j - S_j + T_j$$

$$\bigtriangledown_{b_j} L(w,b,\epsilon,\lambda,v) = E_j - R_j$$

$$\bigtriangledown_{\epsilon_i} L(w,b,\epsilon,\lambda,v) = C - (v_i + \sum_{j=1,j\neq y^i}^{k} \lambda_{ij})$$

Where

$$S_j = \sum_{i=1}^{n}\sum_{p=1,p\neq y^i}^{k} \lambda_{ip}x^i \text{ when } j = y^i$$

$$T_j = \sum_{i}^{n} \lambda_{ij}x^i \text{ when } j \neq y^i$$

$$w_j = \frac{S_j + T_j}{2}$$

$$R_j = \sum_{i=1}^{n}\sum_{p=1,p\neq y^i}^{k} -\lambda_{ip} \text{ when } j = y^i$$

$$E_j = \sum_{i}^{n} \lambda_{ij} \text{ when } j \neq y^i$$

Putting this into our KKT Conditions we get

$$\begin{bmatrix} 2w_j - S_j + T_j \\ E_j - R_j \\ C - (v_i + \sum_{j=1,j\neq y^i}^{k} \lambda_{ij}) \end{bmatrix} = 0$$

With our primal feasibility being

$$\text{st. } w_{y^{(i)}} * x^{(i)} + b_{y^{(i)}} - w_k * x^{(i)} - b_k \geq 1 - \epsilon_i \; \forall \; i, \text{ and for each } k \neq y^{(i)}$$

$$\epsilon_i \geq 0 \; \forall \; i$$

4 Approaches

4.1 Data Preparation

We used the dataset from UCI Machine Learning Repository called Adult Data Set [6]. The dataset had 15 features in which income is dependent and others are independent. There were total of 32561 data points. One important fact about the datatset is that it mostly samples the United States population. Our objective of using this dataset is to classify which of the 3 major occupations a person most likely belongs to. The three major occupations were Administer clerical, Executive manager, and Craft repair.

Out of these 15 features and 32561 data points, we are going to use 11935 many data points and 7 features. The 7 features that we are considering are "education", "education-num", "martial-status", "race", "sex", "hours-per-week", and "salary".

Attribute	Values
Education	HS-grad, Some-college, Bachelors, Masters, Assoc-voc, 11th, Assoc-acdm, 10th, 7th-8th, Prof-school, 9th, 12th, Doctorate, 5th-6th, 1st-4th, preschool
Education num	1-14
marital-status	Married-civ-spouse, Never-married, Divorced, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
race	White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, Other
sex	Male, Female
hours-per-week	1-99
Salary	$\leq 50,000$ or $\geq 50,000$

Table 1: Attributes and Values

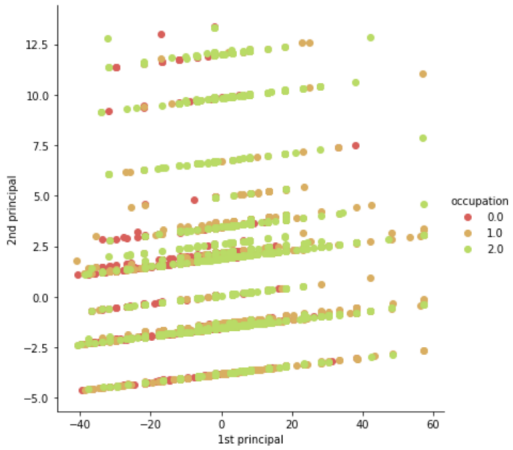

Figure 1: Data Visualization 2D

Here are the visualization of the processed data. Figure 1 gave us an idea of clustering because the same colors form a region when the data was plotted in 2 dimensional space using PCA. But since the data points were overlapping, we plotted a 3D graph instead. Figure 2 clearly shows how data seem clustered into 3 groups by changing the viewing angle.

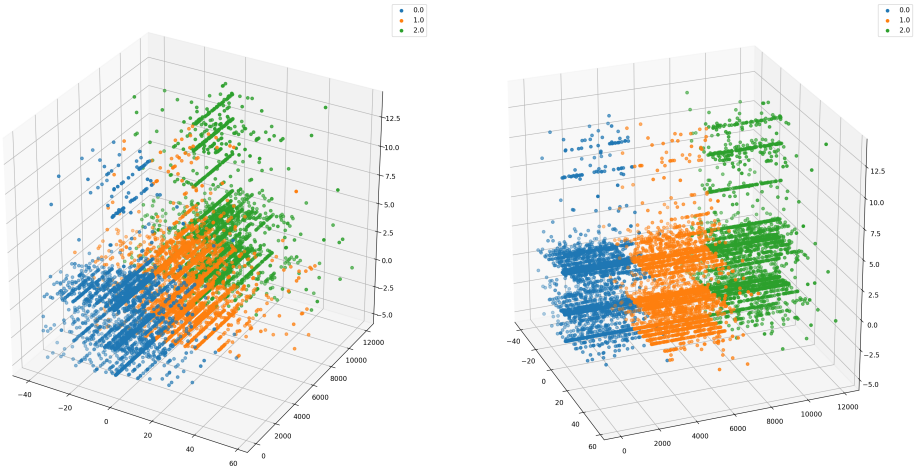

Figure 2: Data Visualization 3D

4.2 Multiclass SVM

We decided Support Vector Machines(SVM) as the base model. It is useful in classifying problems by finding hyperplanes in an N-dimensional space where N is the number of features that distinctly classifies the data points. The hyperplanes would be the boundary that maximizes the distance in between data points that has different labels [7].

Before directly performing SVM to classify 3 different labels, we used PCA to reduce the size of each data points to 3 dimensional space [8]. We wanted to speed up the process of finding the hyperplanes by using PCA. In order to not lose much information from performing PCA, we tried different components from 2 to 5.

Num of Components	Accuracy
2	31.24%
3	**34.0%**
4	32.91%
5	32.56%

Table 2: Accuracy report of different number of components

Previous result Table 2 shows the different results in number of components when all the other conditions are the same. Therefore, for SVM classifier, we decided to use PCA to compress our data to 3 features.In order to optimize our SVM model, we needed to choose different hyper-parameters.

There are mainly 3 things that we needed to decide. Kernel which is the similarity function, parameter C, and parameter gamma. For Kernel, we used the most common Gaussian RBF kernel because it has good approximation capabilities especially for SVM. The C parameter controls the trade off between the classification accuracy of training points and a smooth decision boundary. Thus it suggests the model to choose data points as a support vector. We chose 4 different values for the C parameter. Also, we chose 4 different gamma parameters that defines how far the influence of single training example reaches.

4.3 Random Forest

The random forest classifier is a meta estimator that fits a number of decision tree classifiers for different sub-samples of the provided dataset. Then it uses averaging to improve the prediction accuracy and the randomness of the sub-samples help prevent data over-fitting.

Due to the fact that letting the trees fully grown can lead to very large data set. To reduce memory consumption and complexity, we explored the accuracy of the model in relation to the max depth of the tree.The first model had maximum of 2 layers, the second had 3 layers, third had 4 layers, and the forth had 5 layers. (table). We also tested all these models using different data splits between training and testing to see how different ratios can affect accuracy. More over, we recorded the time it took to train different models.

4.4 Artificial Neural Network

After looking into Random Forest and SVM, we wanted to take a more recent approach in classification techniques. Neural networks are useful as they mimic the human brain, thus allowing for more non-linear situations as well as being able to generalize.[9] Since there might be some underlying rules or some ways that we haven't perceived the data, we thought it would be good to try this approach.

Our input layer into each of these models is 34 nodes, as we transformed the categorical variables into one hot encodings. This is useful because we are able to encode these features as numerical representations that have independence from each other. This is useful as we would not want some values to have more significance than others. This type of encoding method is also commonly used for neural networks.[10] We trained three different models to test the effectiveness of different techniques on this dataset. We ran each model 30 times and recorded the average runtime and accuracy.

For all the models the output layer is 3 nodes that have a softmax activation function. This is because using a softmax activation function will allow for the outputs to produce a probability for each of the three classes (craft-repair, exec-managerial, adm-clerical). Each model was initially trained on 3000 epochs, then modified after to correct for overfitting.

The first model has four different hidden layers (16,32,16,8). Each of these layers uses a sigmoid activation function. Using a sigmoid function might be beneficial as this is a shallow network with a small amount of input nodes. [11] The optimal number of epochs was chosen to be 300.

The second model has four different hidden layers as well (16,32,16,8). The difference is that this model uses the RELU activation function. Most modern neural networks utilize RELU activation functions and are very common. They offer better performance as well as generality compared to the sigmoid function. They also resemble a linear function which makes them easier to optimize.[11] The optimal number of epochs was chosen to be 200.

The final model uses a radial based kernel. The first hidden layer is a gaussian based kernel that transforms the data to 128 dimensions, followed by three hidden layers (64,32,16). A gaussian kernel provides better generality and may be able to represent the data better for this specific task.[12] The hidden layers use a sigmoid activation function as this function performed slightly better than the RELU activation function. The optimal number of epochs was chosen to be 100.

As stated below in the results, the computational complexity was 8.8 seconds for the radial kernel, 13.4 seconds for the RELU activation function, and 21.5 seconds for the sigmoid activation function. These runtimes are based on each models optimal number of epochs. If we were to add more hidden layers and nodes this would increase the computational time. Another factor that would increase the computational time would be the size of the dataset.

5 Results, Conclusion, And Possible Future works

5.1 Results

5.1.1 Multiclass SVM

The result for SVM will be shown in the result of main classification metrics. In order to measure the quality of the prediction made by our SVM model, we stated the precisions, revall, f1-score. These values are computed by using true and false positives(TP, FP), true and false negatives(TN, FN).

Precision shows what percentage of your predictions were correct. It is the ability of classifier to not label an instance positive that is actually negative. It uses the the true positive and false positive values to represent the ratio.

$$precision = \frac{TP}{TP + FP}$$

Recall shows the fraction of positives that were correctly identified. It is the ability of a classifier to find all positive instances. It uses the true possible and false negative values to represent the ratio.

$$recall = \frac{TP}{TP + FN}$$

F1 score shows how balanced it is between precision and recall. It will be a better measure to use if we need to seek a balance between precision and recall and there is an uneven class distribution. It uses the recall and precision values to represent the ratio.

$$f1score = \frac{2 * (recall * precision)}{recall + precision}$$

Table 3 shows the result when we used 90% of the entire data as training set and 10% of the entire data as testing set. We obtained accuracy of 46.57%. Table 4 show the result when we used 95% of the entire data as training set and the rest as testing set. We obtained accuracy of 48.24%. Table 5 show the result when we used 99% of the entire data as training set and the rest as testing set. We obtained accuracy of 46.67%. One thing we can conclude from the result is not that much affected by the size of the dataset that we train on. The more important aspect was normalizing and scaling the data that we already have.

Model	precision	recall	f1-score
Label 0	0.47	0.49	0.48
Label 1	0.43	0.70	0.53
Label 2	0.54	0.16	0.24

Table 3: Classification report of 90% Train data

Model	precision	recall	f1-score
Label 0	0.50	0.53	0.51
Label 1	0.46	0.72	0.57
Label 2	0.64	0.20	0.30

Table 4: Classification report of 95% Train data

Model	precision	recall	f1-score
Label 0	0.35	0.66	0.46
Label 1	0.52	0.32	0.39
Label 2	0.33	0.23	0.27

Table 5: Classification report of 99% Train data

For all 3 different size of training set, the C parameter that resulted in the best was when C was 50 where other C values that we tested were 1, 5, 10. This shows that our model worked better when we tried to overfit the data more. Then we would get the higher variance and lower bias. Also, for the all 3 different size of training set, the gamma parameter that resulted in the best was when gamma

was 0.005 where other gamma values that we tested were 0.0001, 0.0005, 0.001, 0.005. This shows that our model worked better when the gamma is high and when our decision boundary depends on points close to the decision boundary. Also, the points that are near to the decision boundary carries more weights than far away points.

Model	Avg Runtime to converge
90% Train data	47min 38s
95% Train data	4min 29s
99% Train data	5min 20s

Table 6: Training Runtime

Looking at the runtime of each model with different size of dataset, we can see that less data takes much more time to converge. Because training time complexity of SVM depends on number of data points, with the regularization parameter C. [13]. Therefore, we can conclude that 90% training data performed the best when we were testing our SVM model.

5.1.2 Random Forest

Model	90%	95%	99%
Max Depth 2	65.41	66.00	72.50
Max Depth 3	66.00	66.10	72.50
Max Depth 4	66.75	66.16	72.50
Max Depth 5	67.00	66.33	72.50

Table 7: Accuracy Random Forest

As we can see from table 7, random forest performs mostly better when having a higher ratio training data vs test data. The max depth of the tree affect accuracy more when we are using less data to train. Therefore the best model would be 99% training data at max depth 2 which gives the best result with least complexity. According to the feature importance analysis from the model, the gender weighted the most for occupation prediction and the second highest is number of educations, and the third would be salary.

Model	90%	95%	99%
Max Depth 2	208	199	205
Max Depth 3	228	235	260
Max Depth 4	246	251	282
Max Depth 5	260	273	295

Table 8: Training Runtime (milliseconds)

From table 8 we can see that the training time for random forest is fairly short. The more data and deeper of the tree requires longer to train. But overall, it seems much faster than other models.

5.1.3 Artificial Neural Network

Accuracy Using Optimal Epoch

Model	90%	95%	99%
Sigmoid	66.9	66.9	65.5
RELU	67.2	67.0	66.9
Radial	**67.0**	**67.6**	**67.0**

Comparing model accuracy based on training dataset size of original data

Training Runtime (seconds)

Model	Avg Runtime on 3000 epochs	Avg Runtime on optimal epochs
Sigmoid	174.3	21.5
RELU	**173.6**	13.4
Radial	209.7	**8.8**

Based on these accuracy measurements, we can see that each method performs around the same when using their optimal amount of epochs with radial being the front runner. However, based on the runtime we can see a significant margin between the radial based neural network compared to the other two models. This is because the radial based neural network converges to the minimum faster. Overall it would be best to use the radial model out of the three since it has the same accuracy but the lowest training time. This is significantly important for future use of this model on larger datasets and frequent updates.

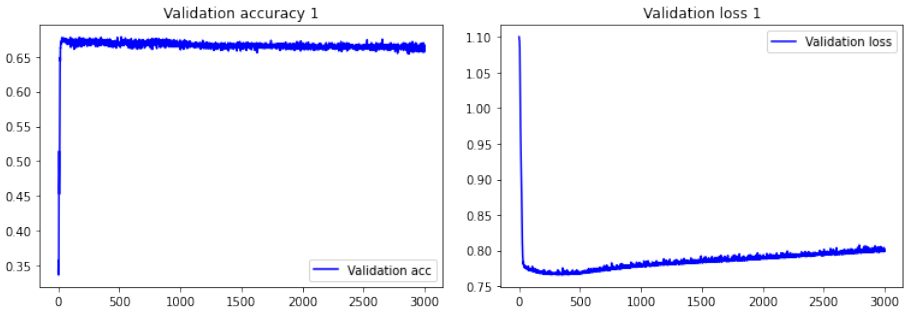

Figure 3: Validation accuracy and loss of sigmoid activation function

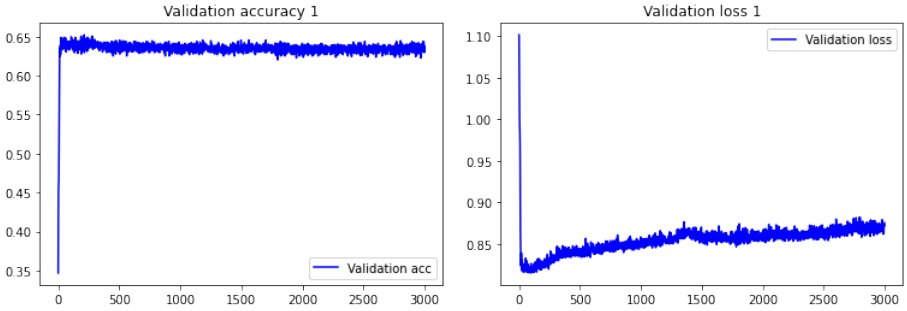

Figure 4: Validation accuracy and loss of RELU activation function

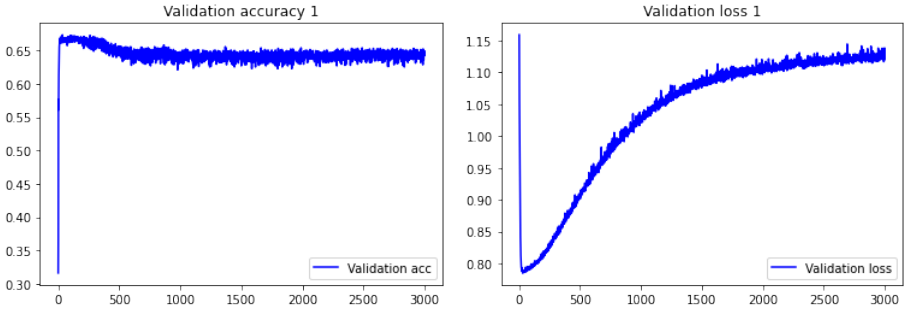

Figure 5: Validation accuracy and loss of radial kernel based neural network

As we can see by the figures above, the optimal amount of epochs would be when the change in the loss function is minimal and the loss function does not start to increase again. From the sigmoid activation function we judged this to be around 300 epochs. For the RELU activation function we judged this to be 200 epochs. For the radial kernel we judged this to be 100 epochs.

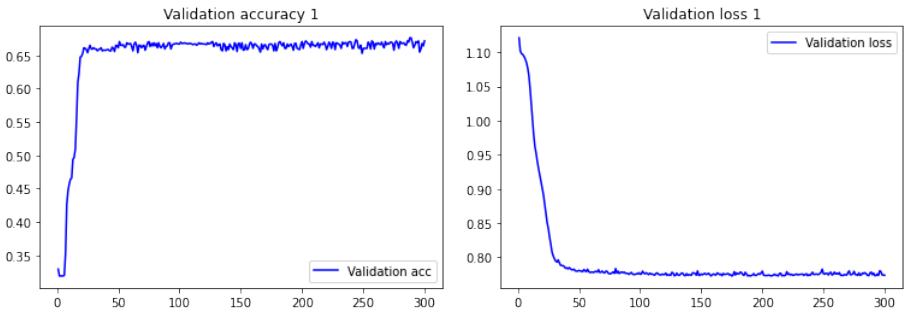

Figure 6: Validation accuracy and loss of sigmoid activation function

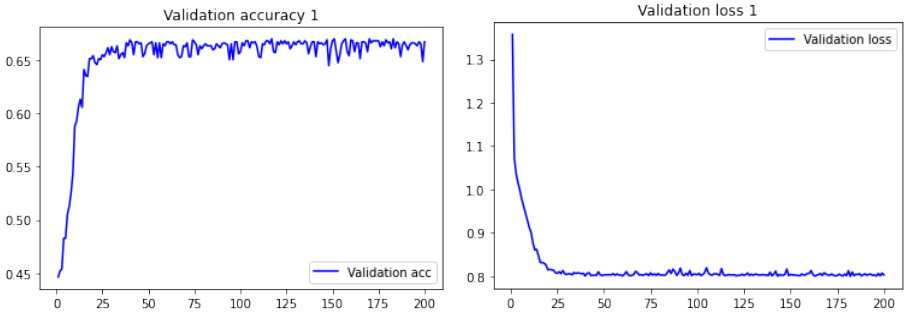

Figure 7: Validation accuracy and loss of RELU activation function

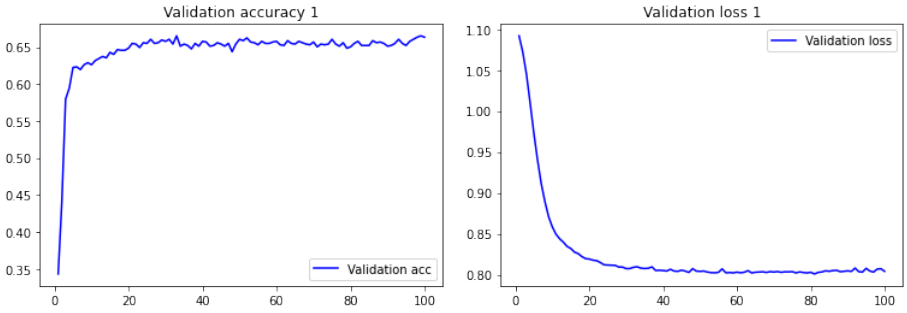

Figure 8: Validation accuracy and loss of radial kernel based neural network

The graphs above show the accuracy and loss function of each model based on their optimal number of epochs. There is a huge different between these graphs and the graphs above, especially in the radial kernel based neural network. This shows that it was appropriate to reduce the number of epochs to minimize overfitting the training data.

5.2 Conclusion

After comparing all the models we decided that it was best to use the random forest classifier for this type of dataset. We reached this conclusion based on the accuracy as well as the runtime. Random

forest and the artificial neural networks both outperformed the multiclass SVM in terms of accuracy and run time. However, random forest has around the same accuracy when trained on 90% and 95% of the data, but has a significant increase in performance when trained on 99% of the data. The main factor on our decision is that random forest has the best runtime by a significant margin. Random forest was able to complete its training about 30x faster than the artificial neural network. This means that we get a similar prediction at a lower cost. This would be beneficial to companies that re-train their model based on new incoming data, as the real world keeps on changing over time.

5.3 Possible Future works

In terms of the neural network model, we would want to improve on this and try different methods that might increase the performance. One paper suggested using a hypersphere classifier as it performs well in creating complex decision regions, which might be beneficial for this dataset.[12] Another method that could be helpful with this model would be changing our encoding scheme for categorical variables, such as polynomial coding.[14]

A possible future application for this is, based on the predicted occupation, to predict the likes, wants, and needs of people with that occupation in order to implement targeted advertising for companies to sell their products to an audience more tailored to that company.

References

[1] Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[2] B. Zadrozny, "Learning and evaluating classifiers under sample," 2004.

[3] M. J. P. Stephen D. Bay, "Detecting group differences: Mining contrast sets. data min. knowl. discov," 2001.

[4] D. Z. T. S. G. Z. L. Z. D. D. M. Thomas Serafini, G. Zanghirati, "Gradient projection methods," 2003.

[5] J. P. Elena Filatova, "Occupation inference through detection and classification of biographical activities," 2012.

[6] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[7] H. Marius, "Multiclass classification with support vector machines (svm), kernel trick amp; kernel functions," Sep 2020.

[8] F. L. Gewers, G. R. Ferreira, H. F. de Arruda, F. N. Silva, C. H. Comin, D. R. Amancio, and L. da F. Costa, "Principal component analysis: A natural approach to data exploration," 2018.

[9] E. Grossi and M. Buscema, "Introduction to artificial neural networks," *European Journal of Gastroenterology Hepatology*, vol. 19, no. 12, p. 1046–1054, 2007.

[10] K. Potdar, T. Pardawala, and C. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, 2017.

[11] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[12] G. Ou, Y. Murphey, and A. Feldkamp, "Multiclass pattern classification using neural networks," *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004.

[13] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[14] K. Potdar, T. Pardawala, and C. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International Journal of Computer Applications*, vol. 175, pp. 7–9, 10 2017.