COGS 118B

Runtime and Performance Analysis for Predicting Income

Final Project

Miguel Garcia
Janina Schuhmacher

UC San Diego

mig053@ucsd.edu
jschuhma@ucsd.edu
fall term 2019
10-12-2019

ABSTRACT

In this project, we compared the performance of different classifiers in a binary classification task. The goal was to predict if a person makes more or less than 50 k per year, given a set of categorical and continuous attributes. The algorithms which we used were support vector machines with linear kernel (SVM) and multilayer perceptrons with different activation functions. Testing accuracy was around 0.86 for all classifiers except for SVM with automated correction for class imbalance which had significantly lower testing accuracy. Training and testing accuracies were similar for all classifiers, indicating good generalization from training to test data. The biggest challenge consisted in the imbalance in the data set, resulting in relatively poor recall rates and, consequently, low F1 scores. Using sklearn's automated correction for class imbalance improved recall but at the cost of lowering precision.

List of Figures

1. Introduction

One of the promises of modern machine learning is that algorithms make sense of large amounts of complex, real-world data, allowing us to explore relations between a set of input data and output data. In our project, we are very interested in exploring how we can use demographic and biographical information about people to predict their economic success. One very relevant related application is predicting credit-worthiness in the banking domain. Another application lies in sociological and psychological research which tries to explore how different demographic and biographical variables affect people's incomes. In this project, we use the "Census Income Data Set" (Dua & Graff, 2017) to train different classifiers. Goal of this task is to predict whether a person earns more or less than 50 k per year. We will train different versions of two very popular Machine Learning algorithms, Support Vector Machines and Neural Network. Both algorithms have received a lot of attention, both in scientific and popular literature, for their ability to learn complex relations in data. In our discussion, we question this hype critically. More generally, we shall compare the performance of different support vector machines and neural networks and discuss implications and difficulties. Additionally, we shall pay particular attention to the problem of unbalanced data and how it affects predictions in our case.

2. Related Work

In the following section, we present general information and related work for the two algorithms we used to classify our data.

2.1 Support Vector Machines

Support vector machines (SVM) are very popular 'off-the-shelf' algorithms (Ng, n.d.). In a recent study which compared the performance of different algorithms on the UCI database, SVM were among the top algorithms (Fernandez-Delgado, Cernadas, Barro, & Amorim, 2014). SVM are supervised machine learning algorithms which can separate non-linearly separable data by embedding them in a higher-dimensional feature space using the kernel-trick. In separating the data, the SVM algorithm tries to maximize the margin between the decision boundary and the data points. Recently, SVM have been employed successfully for tasks such as breast cancer mammography recognition (Azar & El-Said, 2014) or forecasting electricity consumption (Kaytez, Taplamacioglu, Çam, & Hardalac, 2015). Furthermore, similarly to the problem posed in this paper, SVM have been employed to predict whether people make an income above a certain threshold (Guenther & Schonlau, 2016; Lazar, 2004). The results of these studies indicate that SVM are appropriate to analyze the input variables in our dataset with the goal of making predictions about people's income levels.

2.2 Artificial Neural Networks

Artificial neural networks (ANN) are possibly the most hyped machine learning algorithm. In the popular media they are even taken as synonymous with machine learning and artificial intelligence. ANN are non-parametric learning methods that are able to learn very complex, non-linear relations in data. The term artificial neural network includes a wide variety of different algorithms, including for example convolutional neural networks, recurring neural networks, deep reinforcement networks, etc. Popular areas where ANN have recently been employed successfully include natural language processing (e.g. Kalchbrenner, Grefenstette, & Blunsom, 2014) and face recognition (e.g. Li, Lin, Shen, & Brandt, 2015). Related to our problem, ANN have been found to be useful to predict credit risk (e.g. Byanjankar, Heikkilä,

& Mezei, 2015) from variables very similar to the ones in our database, e.g. marital status and age.

2.3 Open questions

Overall, the literature review suggests that SVM and ANN are suitable methods for our prediction problem. In comparing the different classifiers, a good performance of SVM would suggest that the data is linearly separable in a high-dimensional space, indicating that it is not necessary to fit a highly complex ANN. On the other hand, poor performance by SVM versus ANN would indicate that the relation between input and output data requires a more complex model. In our experiment, we shall evaluate in how far the two models yield insightful results and where we see limitations.

3. Method

3.1 Data set

For our project, we use the "Census Income Data Set" (Dua & Graff, 2017). Further information about the data set and relevant publication can be found on the UCI Machine Learning Repository website (`https://archive.ics.uci.edu/ml/datasets/Census+Income`). The classification task which is associated with this data set is to predict whether a person earns more or less than 50k a year. In total, the data set contains data from 32561 persons. This includes 14 partly categorical and partly continuous attributes and, as outcome variable, the information if the person makes more than 50k or less than 50k a year. Figure 1 gives an overview of the raw data set. To train different classifiers on the data set, it is necessary to clean and preprocess the different variables as explained in section 3.2.

Attribute	categorical or continuous	possible values
age	continuous	
workclass	categorical	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt	continuous	
education	categorical	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num	continuous	
marital status	categorical	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	categorical	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	categorical	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	categorical	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	categorical	Female, Male
capital-gain	continuous	
capital-loss	continuous	
hours-per-week	continuous	
native-country		United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands

Figure 1: Attributes in the raw data set.

3.2 Data preprocessing

Figure 2 illustrates the first five instances of the initial data set, including both continuous and categorical variables which are partially redundant and need to be preprocessed before running algorithms. Preprocessing the data entails excluding missing data, choosing which features to use for prediction and one-hot encoding the categorical features.

```
In [2]:
# display df header
data_top = df.head()
data_top
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Figure 2: First five instances of the raw data set.

3.2.1 Excluding missing data

Firstly, we excluded instances with missing data. The new data set consists of 30162 instances.

3.2.2 Choosing appropriate features

Analyzing the feature 'native-country' of the new data set reveals that in 27504 cases, the native country is United States. To avoid any over/underfitting or non-meaningful conclusions due to this imbalance in the data, we excluded all data points with native countries other than the United States.

Furthermore, the feature 'fnlwgt' represents socio-demographic characteristics of the person represented by the data point (see file 'adult.names' on `https://archive.ics.uci.edu/ml/datasets/Census+Incomeforadetaileddescription`). Since we cannot retrospectively retrace the calculation of this feature, we are going to exclude it from our further analysis.

In addition, there are two features describing the persons' education. Since we expect these features to be highly correlated, we only use the feature 'education' for our further analysis, dropping the feature 'education-num'.

This leaves us with 11 from the initially 14 attributes, out of which seven are categorical features (workclass, education, marital-status, occupation, relationship, race and sex) and four are continuous features. Figures 3 to 9 visualize the distribution of values of the seven categorical variables.

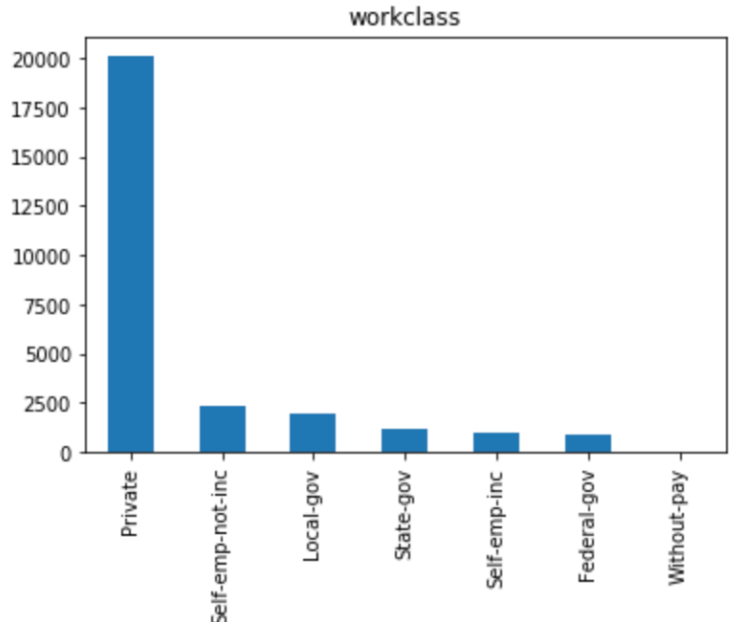

Figure 3: Distribution of the feature 'workclass' in the preprocessed data set.

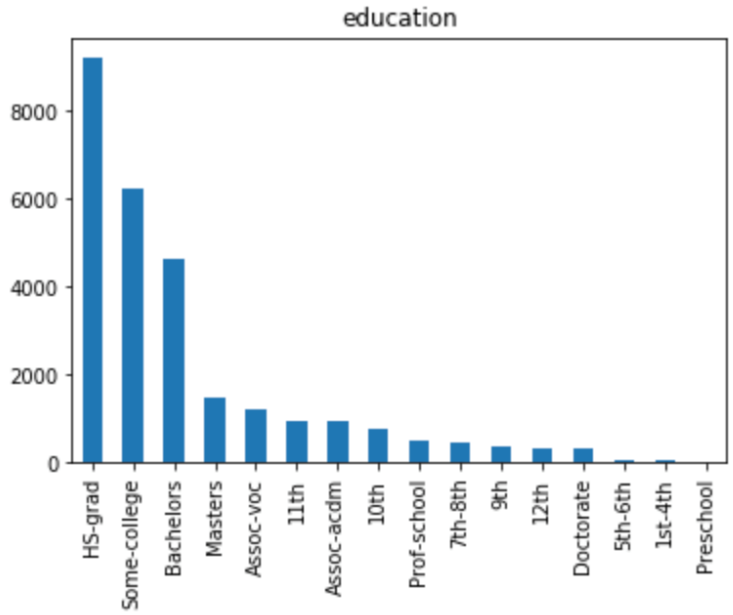

Figure 4: Distribution of the feature 'education' in the preprocessed data set.

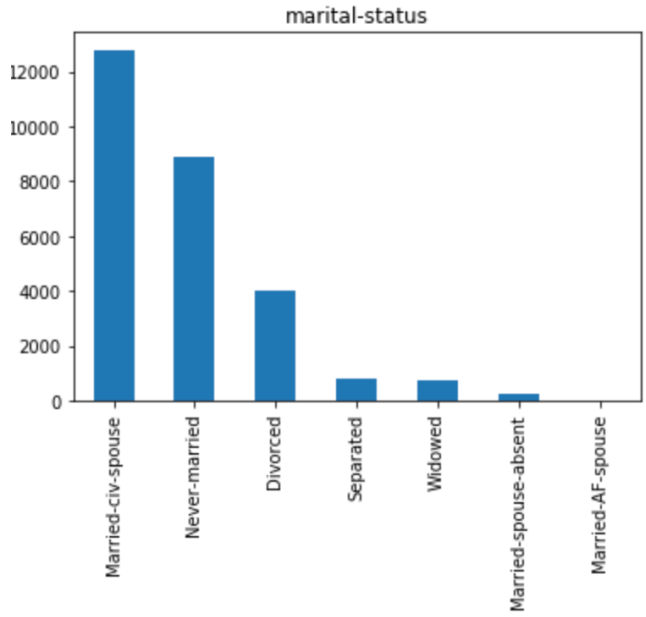

Figure 5: Distribution of the feature 'marital-status' in the preprocessed data set.

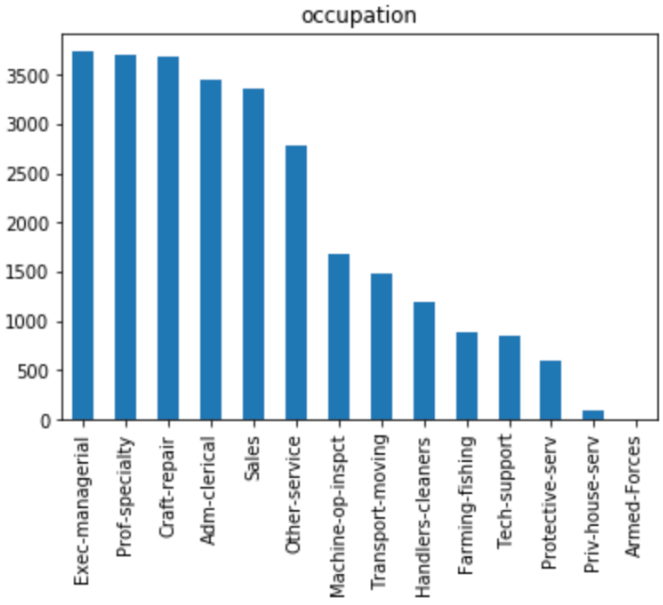

Figure 6: Distribution of the feature 'occupation' in the preprocessed data set.

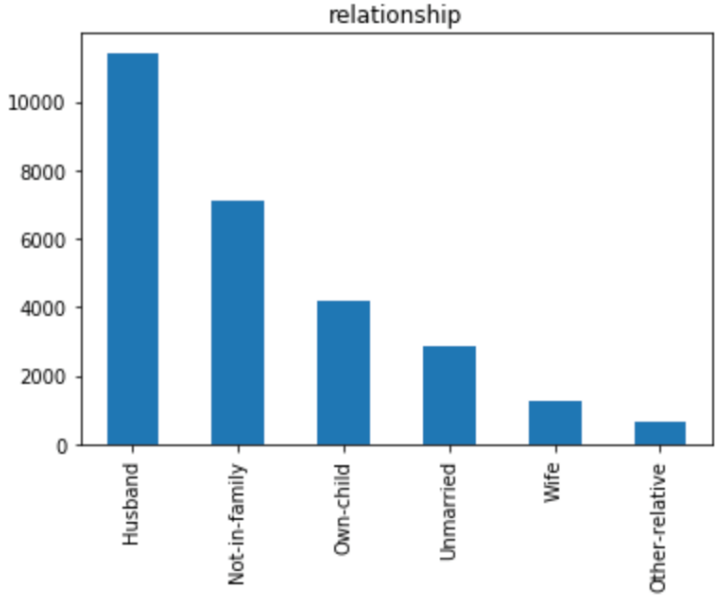

Figure 7: Distribution of the feature 'relationship' in the preprocessed data set.

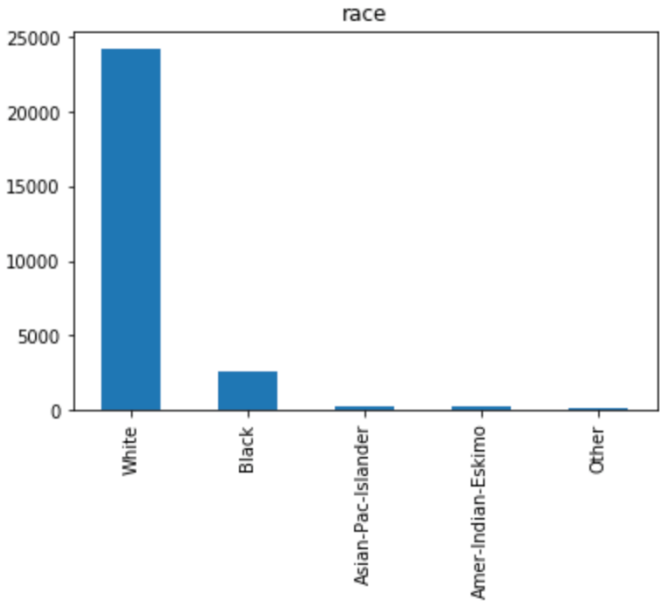

Figure 8: Distribution of the feature 'race' in the preprocessed data set.

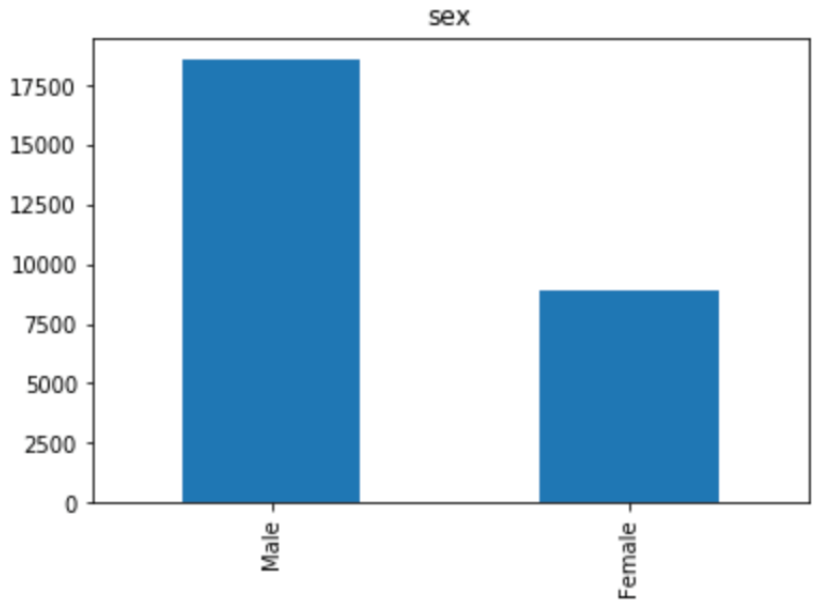

Figure 9: Distribution of the feature 'sex' in the preprocessed data set.

Figures 10 to 13 visualize the distribution of the categorical variables.

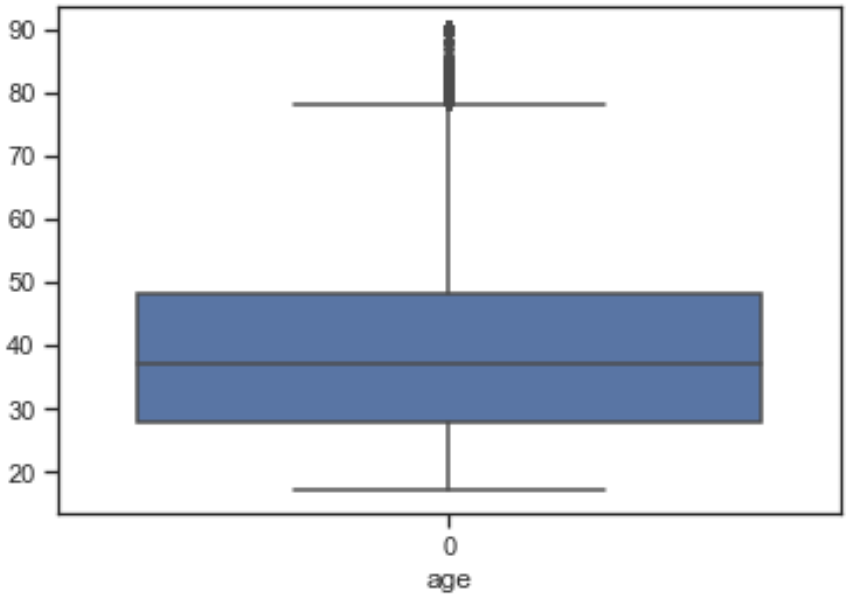

Figure 10: Distribution of the feature 'age' in the preprocessed data set.

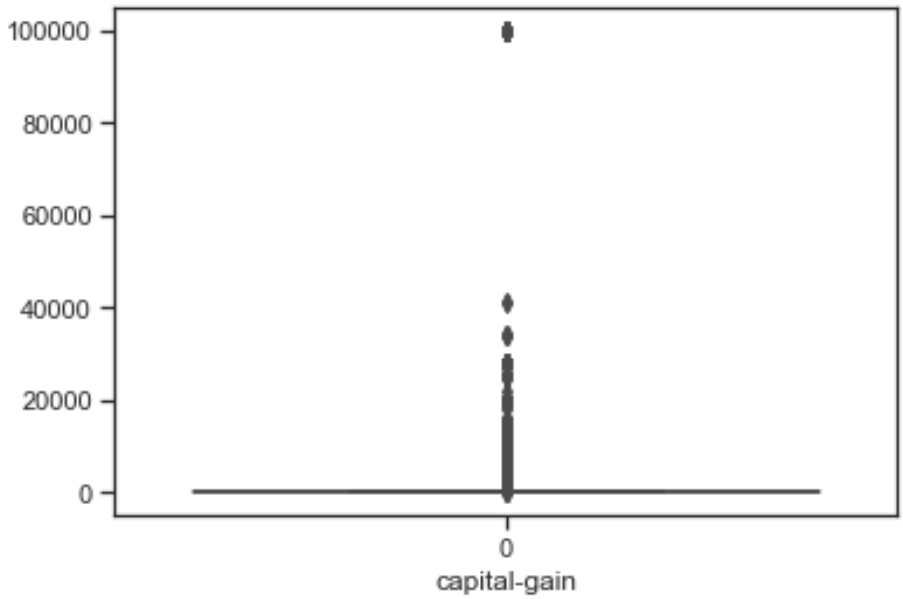

Figure 11: Distribution of the feature 'capital-gain' in the preprocessed data set.

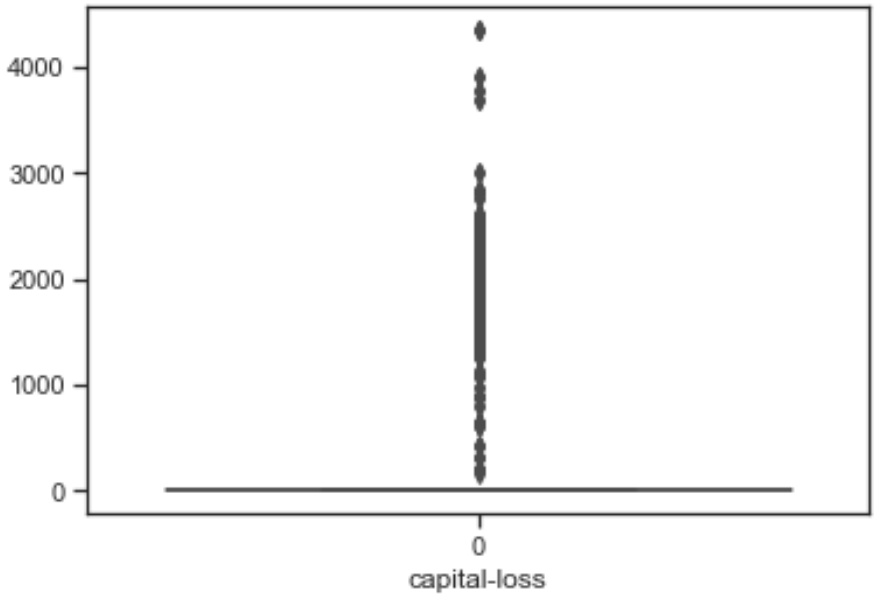

Figure 12: Distribution of the feature 'capital-loss' in the preprocessed data set.

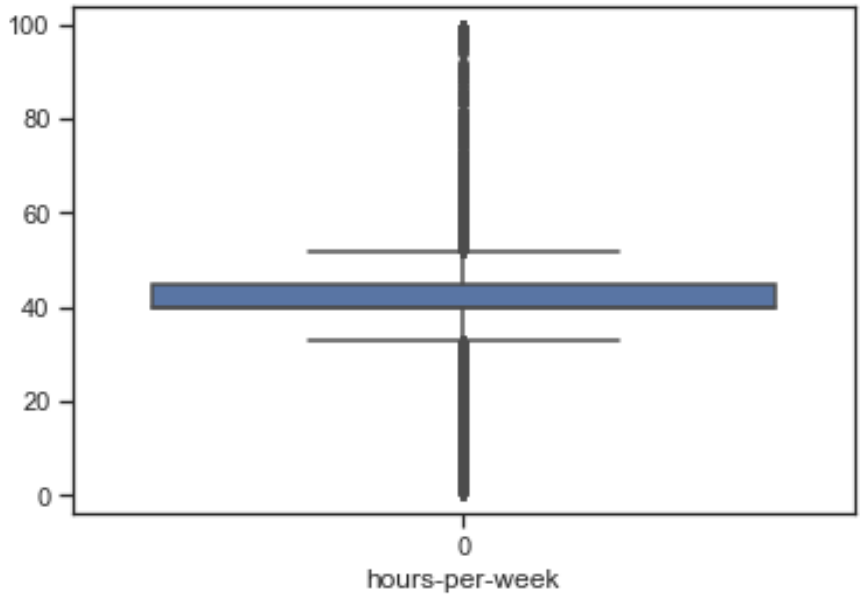

Figure 13: Distribution of the feature 'hours-per-week' in the preprocessed data set.

3.2.3 One-hot encoding of the categorical variables

Using the pandas library, we one-hot encode the categorical variables. As a result, we obtain one variable with values 1 and 0 for each possible value of each categorical attribute. For the final label, we also use 0 and 1 to represent income less than 50k and more than 50k respectively.

3.3 Description of the final data set

After preprocessing the data as described in the previous section, we keep 27504 instances which are described through 45 columns. Figure 14 shows the name of all columns. It is important to note that the data set is highly unbalanced: 20509 of the persons in the data point earn less than or equal to 50k, whereas only 6995 of the persons in the data set earn more than 50k. This means we have 2.93 times more data points for class 0 than for class 1. In the following sections, we shall pay particular attention to this imbalance and discuss implications.

```
In [37]:  1 new_df.columns[1:46]

Out[37]: Index(['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss',
                'hours-per-week', 'workclass_Federal-gov', 'workclass_Local-gov',
                'workclass_Private', 'workclass_Self-emp-inc',
                'workclass_Self-emp-not-inc', 'workclass_State-gov',
                'workclass_Without-pay', 'marital-status_Divorced',
                'marital-status_Married-AF-spouse', 'marital-status_Married-civ-spouse',
                'marital-status_Married-spouse-absent', 'marital-status_Never-married',
                'marital-status_Separated', 'marital-status_Widowed',
                'occupation_Adm-clerical', 'occupation_Armed-Forces',
                'occupation_Craft-repair', 'occupation_Exec-managerial',
                'occupation_Farming-fishing', 'occupation_Handlers-cleaners',
                'occupation_Machine-op-inspct', 'occupation_Other-service',
                'occupation_Priv-house-serv', 'occupation_Prof-specialty',
                'occupation_Protective-serv', 'occupation_Sales',
                'occupation_Tech-support', 'occupation_Transport-moving',
                'relationship_Husband', 'relationship_Not-in-family',
                'relationship_Other-relative', 'relationship_Own-child',
                'relationship_Unmarried', 'relationship_Wife',
                'race_Amer-Indian-Eskimo', 'race_Asian-Pac-Islander', 'race_Black',
                'race_Other', 'race_White'],
               dtype='object')
```

Figure 14: Name of columns of the cleaned data set

3.4 Algorithms and Libraries used

We use several packages for this paper. To read and preprocess the data we use csv (*csv — CSV File Reading and Writing*, 2019), NumPy (Oliphant, 2006) and Pandas (McKinney, 2010). For

visualizations, we use matplotlib.pyplot (*matplotlib version 3.1.1*, 2019). To measure the runtime, we use the package Time (*Time access and conversions*, 2019). To standardize the data, we use sklearn's standard scaler package (*sklearn.preprocessing.StandardScaler*, 2019). We use sklearn's implementation of Dummy Classifiers (*sklearn.dummy.DummyClassifier*, 2019), Support Vector Machines (SVM) (*1.4 Support Vector Machines*, 2019) and Multi-Layer-Perceptron Classifier (MLP) ($sklearn.neural_network.MLPClassifier$, 2019) for the Neural Network. To perform cross-validation when testing the algorithms, we also use sklearn's implementation of cross-validation scoring (*3.1. Cross-validation: evaluating estimator performance*, 2019). Most of the machine learning applications use sklearn's library (Pedregosa et al., 2011).

3.5 Training the classifiers

Classifiers we will use with different initializations

```
#classifiers we will use with different initializations
classifiers = [SVC(kernel = 'linear',gamma ='auto',random_state = 50)]
classifiers.append(SVC(kernel = 'linear',gamma ='auto',random_state = 50,class_weight = 'balanced'))
classifiers.append(MLPClassifier(activation = 'relu',solver='lbfgs', alpha=10,hidden_layer_sizes=(4, 2)))
classifiers.append(MLPClassifier(activation = 'logistic',solver='lbfgs', alpha=10,hidden_layer_sizes=(4, 2)))
classifiers.append(MLPClassifier(activation = 'tanh',solver='lbfgs', alpha=10,hidden_layer_sizes=(4, 2)))
classifiers.append(DummyClassifier(strategy="most_frequent")) #"most_frequent": always predicts the most
##frequent label in the training set.
```

Figure 15: Classifiers used for the experiment and their initializations

For this project, we trained six different classifiers:

1. SVM with Linear Kernel
2. SVM with Linear Kernel and balanced class weight
3. Neural Network with ReLU activation
4. Neural Network with Logistic Sigmoid activation
5. Neural Network with tanh activation
6. Dummy Classifier that predicts most frequent class

Firstly, we split the data into a training set which contains 80 percent of the data and a test set which contains the remaining 20 percent. Secondly, we fit the different classifiers to the training set, using individual attributes and finally all attributes. Thirdly, we measured the time it takes for each classifier to train as well as accuracy on the training set. Lastly, we

computed testing accuracy, F1 scores and ROC curves to evaluate each classifier's performance on the test set.

Figure 15 illustrates the different classifiers and their initializations. For the SVM algorithm, we chose linear Kernel in both versions of the classifier, set gamma to 'auto' and 'random_state' to 50. Initially, we also trained SVM with radial basis function and sigmoid kernels. However, these two classifiers did not perform better than the SVM with linear kernel (see Appendix B). Due to the limited scope of the project, we decided to to only include linear SVM in our final analysis. Linear SVM penalize the intercept whereas a regular SVM does not. By using Linear SVM we also do one-vs-rest rather than one-vs-one with SVM. Most importantly, due to the size of the data, using a Linear SVM is more practical due to its fast convergence with large amounts of data. We compare this with a balanced class weight Linear SVM.

As to the ANN, one of the critical parameters to choose is the activation function. We trained three different multi-layer perceptrons, using rectified linear unit function, logistic sigmoid function and hyperbolic tan functions activation functions. Other than that, we kept the L2 penalty (regularization term) parameter constant at 10 for each version of the classifier and the number of hidden units at two with four and two neurons, respectively.

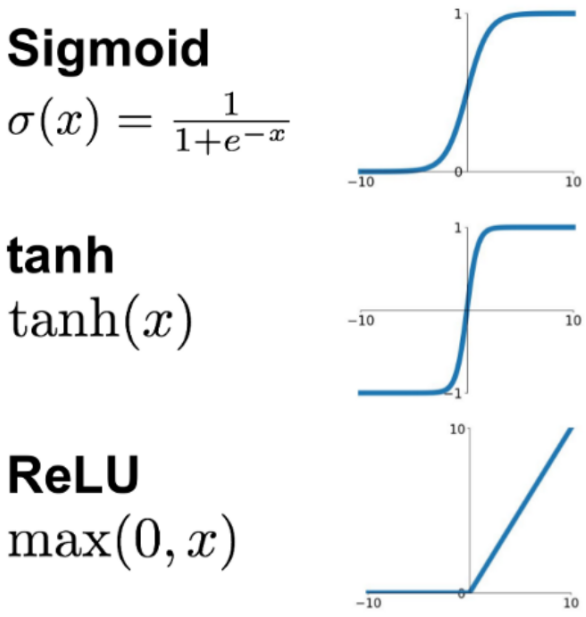

Figure 16: Visualization of different activation functions for ANN (Jadon, 2018)

4. Results

In this section, we shall discuss what we discovered when comparing training the classifiers with different predictor variables, analyzing training and test accuracies, run times and recall versus precision.

4.1 Comparing testing accuracies

To begin with, we computed test accuracy of all classifiers since accuracy is one of the most frequent measures to evaluate a classifier's performance (*Classification: Accuracy*, 2019). Figure 17 visualizes testing accuracy for all classifiers, using either individual attributes as predictors or all attributes. In the plot, the constant brown line shows the testing accuracy of the dummy classifier which we shall use as our baseline. Due to the imbalance in our data set, the testing accuracy of the dummy classifier is relatively high (around 0.75). Interpreting the other results keeping this in mind, we can see that, across different classifiers, the attributes age, hours per week, marital status, occupation, relation-ship and race are useless as single predictors since they do not increase accuracy above baseline. For all classifiers except the neural network with ReLU activation, capital gain is the best single predictor in the test run displayed in Figure 17 (resulting in test accuracies of about 0.79). For the variables, the picture is less clear.

The overall testing accuracy when all available information is used for prediction is comparatively high for all classifiers (around 0.85) except the SVM with balanced class weight which resulted in slightly lower testing accuracy (around 0.81). The fact that single predictors did not increase test accuracies significantly, whereas using all available predictors did, could be due to interactions between the variables and a more complex relation between the given attributes and the outcome variable.

In the following, we shall therefore use all variables for prediction. The overall results indicate that all classifiers achieve to similar test accuracy, except the SVM with balanced class weights which performs slightly worse.

Figure 17: Test accuracies per variable for all classifiers

4.2 Comparing runtime for different classifiers

Figure 18 visualizes the runtime for training the different algorithms. Clearly, the support vector machines take a lot more time for training than the neural networks (about ten times as long). However, in our case, the absolute times are still reasonably fast even for the slowest classifier (around 44 seconds). In addition, the long training time for support vector machines is in part offset by the fact that SVM are very quick when it comes to making predictions.

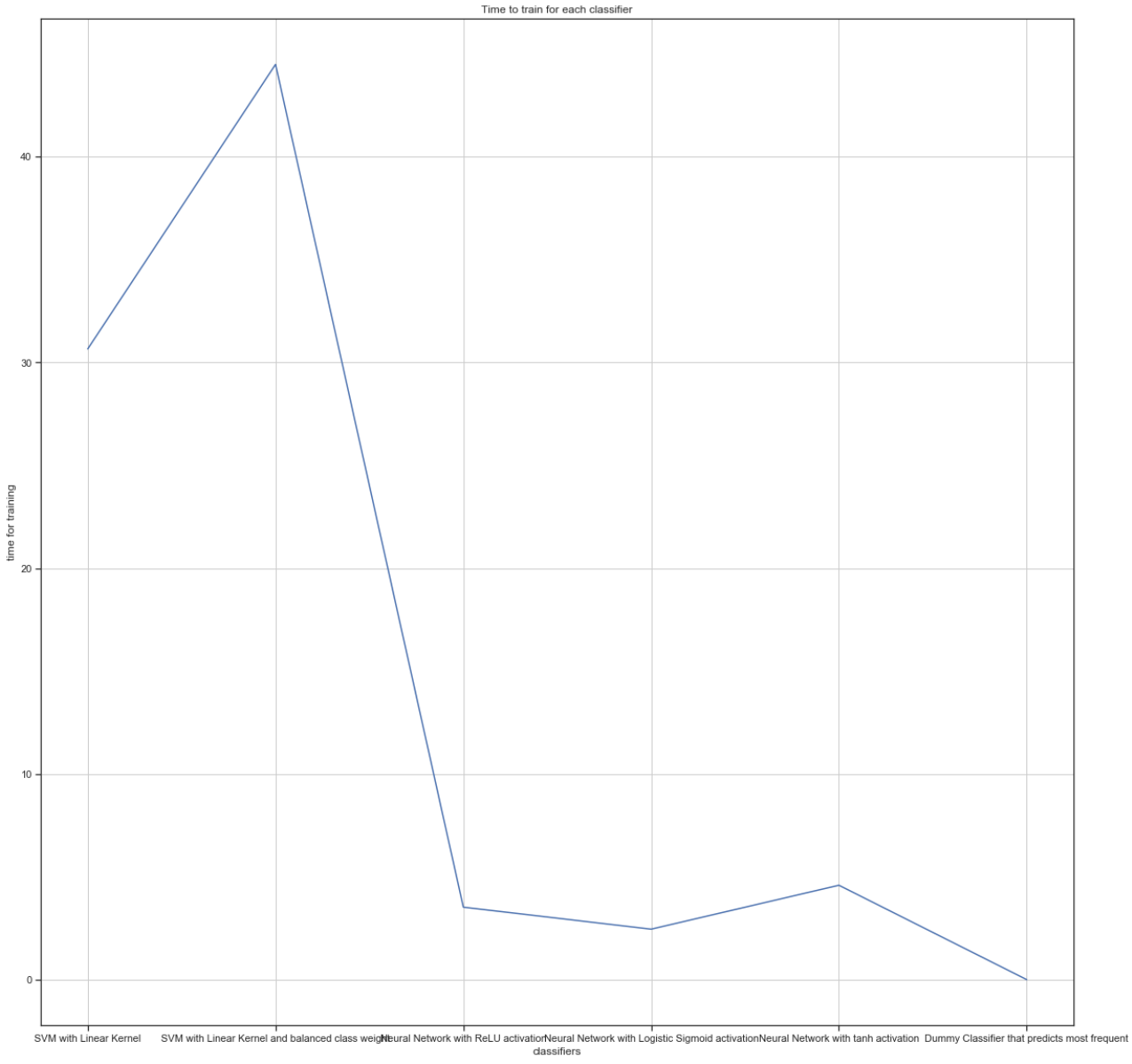

Figure 18: Runtimes for all classifiers

4.3 Comparing training versus test accuracies

Figure 19 plots the accuracies of the different classifiers for the training versus the test set. For all classifiers, test and training accuracies are very similar. This shows that none of the classifier overfits the data. In other words, the classifiers generalize well from the training data set to the test set.

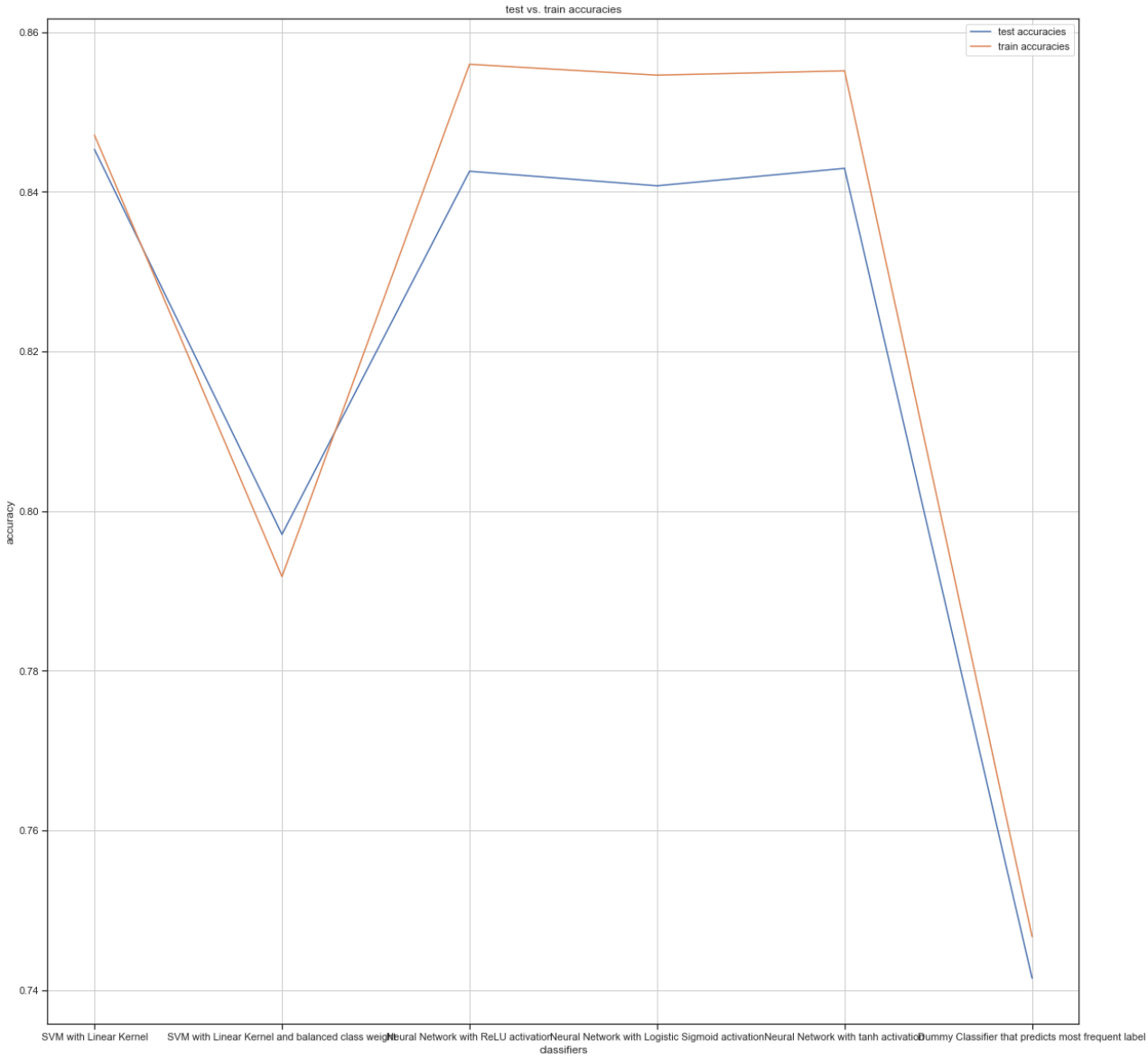

Figure 19: comparison of training and test accuracies for all classifiers

4.4 ROC curves and F1 scores

As noted above, the fact that our data set is very imbalanced can impact the meaningfulness of testing accuracy as a measure of performance. It is therefore important to additionally look at precision and recall of the classifiers.

Since we only work with two outcome variables, we can phrase our prediction task in terms of a binary classification problem, where the goal is to predict if a person earns more than 50 k. Hence, a true positive is when the algorithm correctly predicts that a person earns more than 50 k. A false positive is when the classifier predicts that the person earns more than 50 k even though they actually earn less. Analogously, a true negative is a case in which the classifier correctly predicts that a person earns less than 50 k and a false negative is when the classifier predicts that a person earns less than 50 k even though they earn more.

In this context, precision is defined as the proportion of true positive to all positive predictions (*Classification: Precision and Recall*, 2019):

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

Recall is defined as the proportion of all data points which are classified as positive out of all positive cases (*Classification: Precision and Recall*, 2019):

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

Inherently, recall and precision present a trade-off. We shall therefore also analyze the F1 score, which combines precision and recall (Saito & Rehmsmeier, 2015). The F1 score is defined as the harmonic mean between precision and recall.

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

In our case, since we want to avoid False Negatives and False Positives equally, this is a good measure. If the F1 score close to 1, this means that both precision and recall are high. In Figure 20, we can see that for the classifiers which we trained, the F1 scores are all close to 0.65 (except for the Dummy classifier which naturally has an F1 score of 0). No classifier outperforms the others and no classifier is a lot worse than the others.

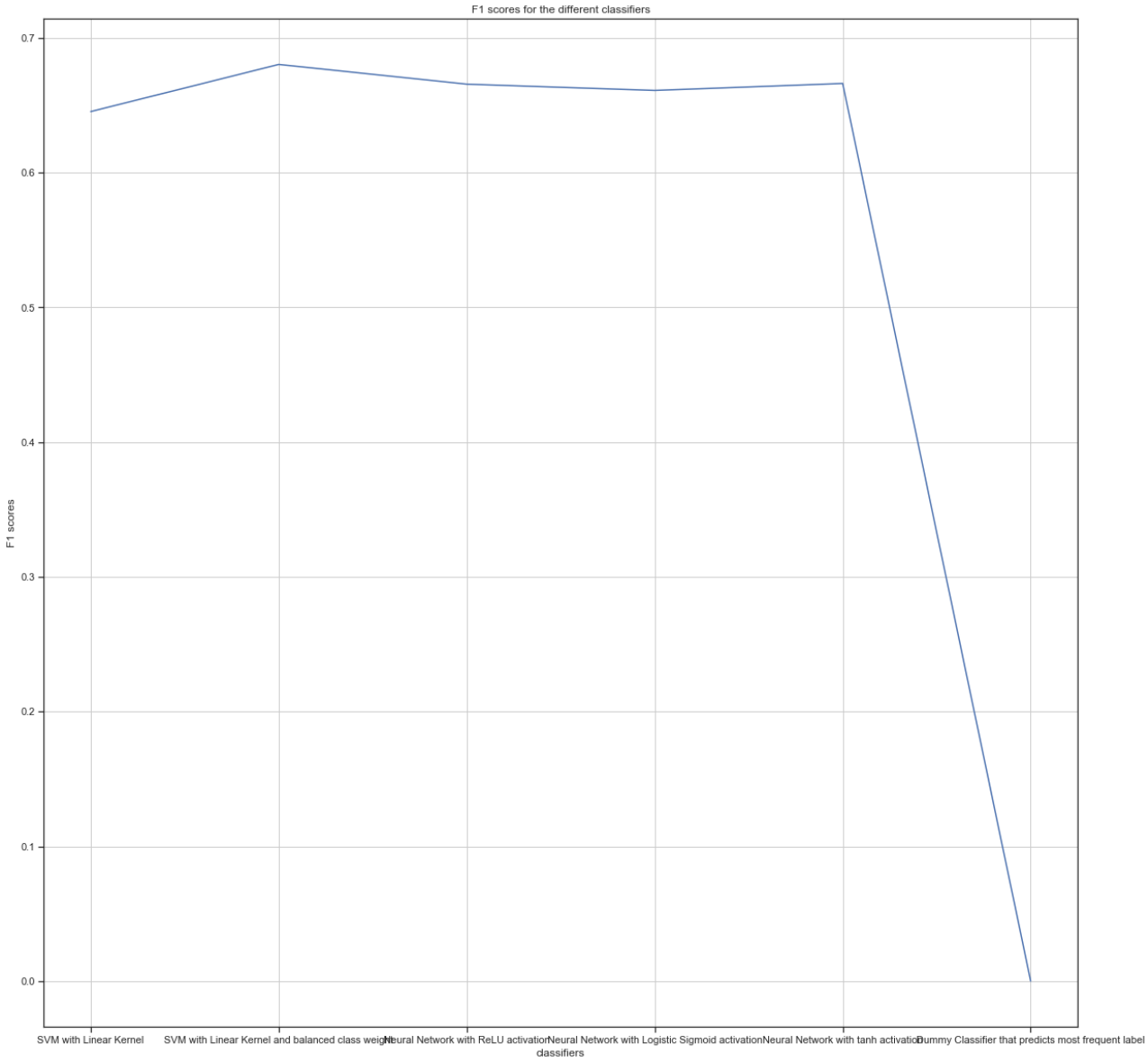

Figure 20: F1 scores for all classifiers

To analyze these results further, we look at the confusion matrices. Figure 21 shows the confusion matrix for the ANN with ReLU activation and figure 22 shows the confusion matrix for the SVM classifier without correcting for class imbalance. For the ANN, we get 837 true positives, 3844 true negatives, 300 false positives and 520 false negatives. The striking result is that out of the 1357 data points with true outcome '$>$ 50 k', 520 are falsely classified as '$<$ 50 k'. This leads to a relatively low recall score of

$$\frac{837}{837 + 520} = 0.617$$

and a precision of

$$\frac{837}{837 + 300} = 0.736$$

This also reflects in the ROC curve: The area under the curve (AUC) is not very large for any of the classifiers, meaning that to achieve high true positive rates, we have to accept high false positive rates (*Classification: ROC Curve and AUC*, 2019). However, since we have highly imbalanced data, in our case the F1 score is the more appropriate and interesting measure (Saito & Rehmsmeier, 2015). The picture looks similar for the other classifiers. For example, the SVM has a recall rate of 0.581 and a precision rate of 0.728. Confusion matrices for the other classifiers can be found in Appendix A.

```
Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3844        300
True >50k        520        837

F1-score is: 0.6712109061748196
```

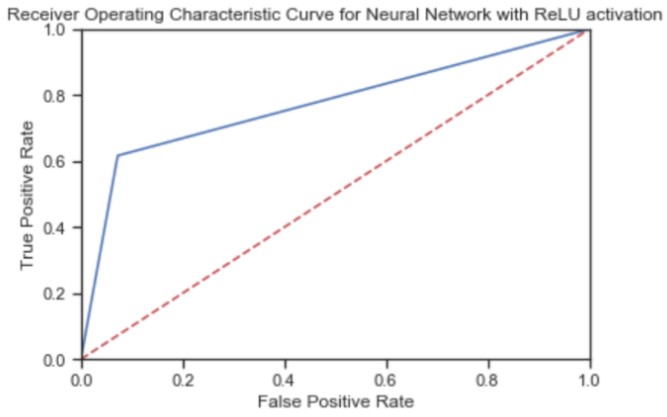

Figure 21: F1 score and ROC curve for ANN with ReLU activation

```
Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3850        294
True >50k        569        788

F1-score is: 0.6461664616646167
```

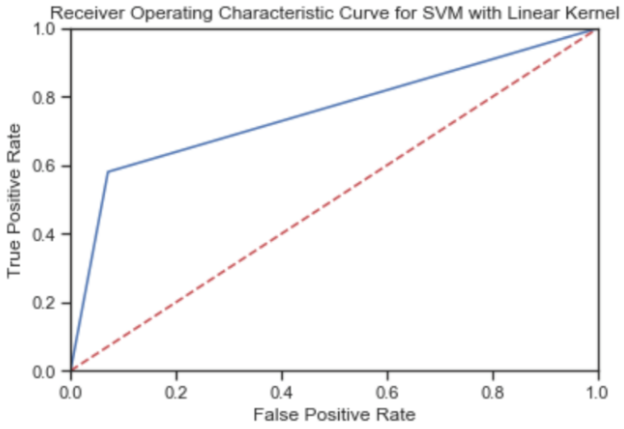

Figure 22: F1 score and ROC curve for SVM without correction for class imbalance

4.5 Dealing with class imbalance

Since our previous results indicate that the imbalance in our data affects the classifiers' performances negatively, we also trained one SVM using sklearn's automatic correction for unbalanced data (*SVM: Separating hyperplane for unbalanced classes*, 2019) . Surprisingly, the classifier which corrected for class imbalance had the lowest accuracy (see Figure 17). When looking at the F1 score, the confusion matrix and the ROC curves (Figure 23), the F1 score is 0.69 (versus 0.65), the precision rate is 0.575 (versus 0.728) and the recall rate is 0.873 (versus 0.581). We can conclude from this that the correction improves recall at the cost of decreasing precision rate. Overall, the F1 score is still not significantly better than the F1 scores of the other classifiers. Therefore, it seems that sklearn's automatic correction for unbalanced data is not the optimal solution for our prediction problem. To improve overall performance, further options should be explored. However, this is out of scope for this class project.

```
Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3159         919
True >50k        181        1242

F1-score is: 0.6930803571428572
```

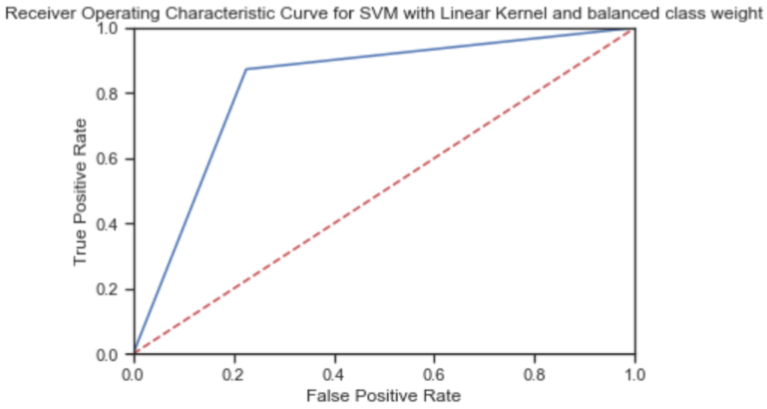

Figure 23: F1 score and ROC curve for SVM with correction for class imbalance

5. Discussion

Overall, our results indicate that in this prediction task, none of the attributes had very high predictive values on its own. Therefore, we used all available information for training the classifiers. Comparing run times for the different classifiers indicates that neural networks are a lot faster to train than support vector machines. However, this is in part offset by the fact that support vector machines make predictions very quickly and that the absolute runtime was not too high for any classifier. For all our classifiers, training and testing accuracy are relatively similar to each other. This is positive, meaning that the classifiers generalize well to unseen data and that there is no overfitting. Besides, the test accuracies were similar for all classifiers, indicating that SVM and ANN capture the relations in the data equally well. This is interesting, since ANN and SVM have different ways to compute decision boundaries. However, the absolute scores are not very high.

5.1 Potential extensions to our project

One explanation for the relatively low test accuracies could be that the available attributes do not predict the outcome variable very well. Another explanation would be that the classifiers were not optimal for fitting the given data, either because they were not complex enough or because we did not choose the optimal model for the given problem. One promising extension to our project would be to train more complex neural networks, e.g. with more hidden layers and neurons, to explore if there is a causal relationship in the data that we just could not model with the classifiers which we have employed in the class project.

Looking at the confusion matrizes and precision versus recall scores, it becomes clear that the low recall is one of the main problems of this classification task. This points us to a further problem, namely the low number of data points for the class 'income $>$ 50 k'. One option to cope with this problem has been explored by using sklearn's automatic correction for SVM classifiers which penalizes misclassifications for the underrepresented class more strongly. Thus, we were able to improve recall at the cost of worse precision without achieving a high F1 score. Since, in this context, recall and precision are equally important measures, another interesting extension to the project would be to try and optimize recall and precision rates through oversampling. This is a method frequently used in Machine Learning to counter the effects of imbalanced data. One promising technique which could be used is Synthetic Minority

Over-sampling Technique (Fernández, García, Herrera, & Chawla, 2018).

5.2 Conclusion

Overall, our results show that even though SVM and ANN are very hyped in the popular and scientific literature, they provide no miracle cure for any machine learning problem. In our experiment, performance of the classifier was limited by the number of data points and the imbalance in the data. We suggest training a more complex ANN and using oversampling as ways to potentially improve performance. At the same time, we need to explore to what extent performance can even be improved. The predictive qualities of the given attributes also limit performance naturally, so that it simply may not be possible to predict income any more accurately, given the attributes in the data set.

6. Python Code and Files

We include .pdf and .ipynb files in the following github link that contain all the output and python code that we used to clean the data, adjust the algorithm parameters, and all the code that we use to train datasets with separate variables, and also includes console output. Note that the console output is different when running the code various times since shuffling the data makes the algorithms train slightly different. https://github.com/miguelgd54/118BFinal-Project

References

1.4 support vector machines. (2019). Retrieved 12-08-2019, from https://scikit-learn.org/stable/modules/svm.html

3.1. cross-validation: evaluating estimator performance. (2019). Retrieved 12-08-2019, from https://scikit-learn.org/stable/modules/cross_validation.html

Azar, A. T., & El-Said, S. A. (2014, Apr 01). Performance analysis of support vector machines classifiers in breast cancer mammography recognition. *Neural Computing and Applications*, *24*(5), 1163–1177. Retrieved from https://doi.org/10.1007 doi: 10.1007/s00521-012-1324-4

Byanjankar, A., Heikkilä, M., & Mezei, J. (2015, 12). Predicting credit risk in peer-to-peer lending: A neural network approach.. doi: 10.1109/SSCI.2015.109

Classification: Accuracy. (2019). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/accuracy

Classification: Precision and recall. (2019). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

Classification: Roc curve and auc. (2019). Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

csv — csv file reading and writing. (2019). Retrieved 12-08-2019, from https://docs.python.org/3/library/csv.html

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from http://archive.ics.uci.edu/ml

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018, January). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *J. Artif. Int. Res.*, *61*(1), 863–905. Retrieved from http://dl.acm.org/citation.cfm?id=3241691.3241712

Fernandez-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014, 10). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, *15*, 3133-3181.

Guenther, N., & Schonlau, M. (2016). Support vector machines. *The Stata Journal*, *16(4)*.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014, 04). A convolutional neural network for modelling sentences. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, *1*. doi: 10.3115/v1/P14-1062

Kaytez, F., Taplamacioglu, M., Çam, E., & Hardalac, F. (2015, 05). Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines. *International Journal of Electrical Power & Energy Systems*, *67*. doi: 10.1016/j.ijepes.2014.12.036

Lazar, A. (2004, 01). Income prediction via support vector machine. In (p. 143-149). doi: 10.1109/ICMLA.2004.1383506

Li, H., Lin, Z., Shen, X., & Brandt, J. (2015, 06). A convolutional neural network cascade for

face detection. In (p. 5325-5334). doi: 10.1109/CVPR.2015.7299170

matplotlib version 3.1.1. (2019). Retrieved 12-08-2019, from `https://matplotlib.org`

McKinney, W. (2010). Data structures for statistical computing in python. In (Vol. 445, p. 51–56). Retrieved from `https://pandas.pydata.org/`

Ng, A. (n.d.). *Cs229 lecture notes. part v. support vector machines.* Retrieved from `http://cs229.stanford.edu/notes/cs229-notes3.pdf`

Oliphant, T. E. (2006). *A guide to numpy* (Vol. 1). Trelgol Publishing USA. Retrieved 12-08-2019, from `https://numpy.org`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *jmlr*, 28252830. Retrieved from `http://www.jmlr.org/papers/v12/pedregosa11a.html`

Saito, T., & Rehmsmeier, M. (2015, 03). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*, e0118432. doi: 10.1371/journal.pone.0118432

sklearn.dummy.dummyclassifier. (2019). Retrieved 12-08-2019, from `https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html`

$sklearn.neural_network.mlpclassifier.(2019).Retrieved12-08-2019, from$

sklearn.preprocessing.standardscaler. (2019). Retrieved 12-08-2019, from `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

Svm: Separating hyperplane for unbalanced classes. (2019). Retrieved 12-08-2019, from `https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane_unbalanced.html`

Time access and conversions. (2019). Retrieved from `https://docs.python.org/3/library/time.html`

A. Results for all classifiers

```
---------- Results Summary for Neural Network with ReLU activation for the column: All Columns ----------

1.4526197910308838 seconds for  Neural Network with ReLU activation  to train
Using All Columns as a predictor for  Neural Network with ReLU activation  we get train accuracy of:  0.8538835613325
456
Using All Columns as a predictor for  Neural Network with ReLU activation  we get test accuracy of:  0.85093619341937
83

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3844         300
True >50k        520         837

F1-score is: 0.6712109061748196
```

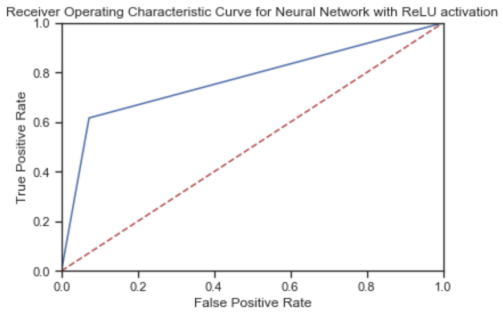

Figure 24: Results for the ANN with ReLU activation

```
---------- Results Summary for Neural Network with Logistic Sigmoid activation for the column: All Columns ----------

1.1201210021972656 seconds for  Neural Network with Logistic Sigmoid activation  to train
Using All Columns as a predictor for  Neural Network with Logistic Sigmoid activation  we get train accuracy of:  0.8
569286006453666
Using All Columns as a predictor for  Neural Network with Logistic Sigmoid activation  we get test accuracy of:  0.84
76640610798036

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3817        353
True >50k        485        846

F1-score is: 0.6687747035573124
```

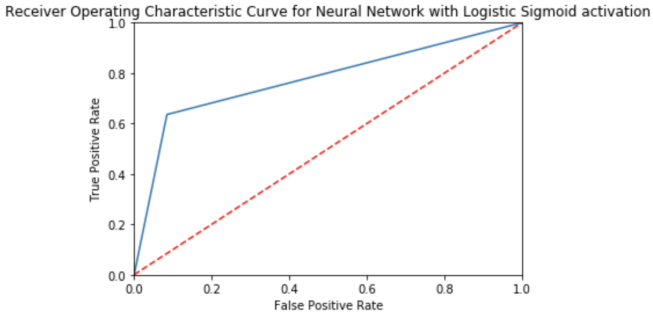

Figure 25: Results for the ANN with sigmoid activation

```
---------- Results Summary for Neural Network with tanh activation for the column: All Columns ----------

1.4165029525756836 seconds for  Neural Network with tanh activation  to train
Using All Columns as a predictor for  Neural Network with tanh activation  we get train accuracy of:  0.8569286006453
666
Using All Columns as a predictor for  Neural Network with tanh activation  we get test accuracy of:  0.84893655698963
82

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3839        331
True >50k        500        831

F1-score is: 0.6666666666666666
```

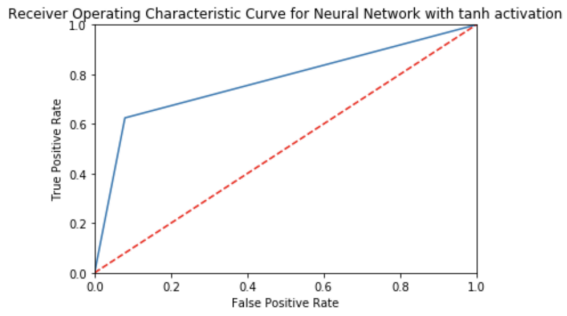

Figure 26: Results for the ANN with tanh activation

```
---------- Results Summary for SVM with Linear Kernel for the column: All Columns ----------

24.022803783416748 seconds for  SVM with Linear Kernel  to train
Using All Columns as a predictor for  SVM with Linear Kernel  we get train accuracy of:  0.8461573421806118
Using All Columns as a predictor for  SVM with Linear Kernel  we get test accuracy of:  0.85

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3850        294
True >50k        569        788

F1-score is: 0.6461664616646167
```

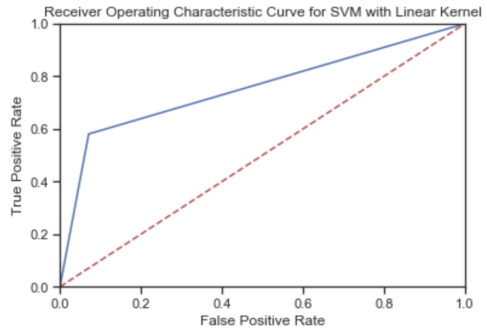

Figure 27: Results for SVM with linear kernel without correction for class imbalance

```
---------- Results Summary for SVM with Linear Kernel and balanced class weight for the column: All Columns ---------
-

30.424149990081787 seconds for  SVM with Linear Kernel and balanced class weight  to train
Using All Columns as a predictor for  SVM with Linear Kernel and balanced class weight  we get train accuracy of:  0.
7898013907194473
Using All Columns as a predictor for  SVM with Linear Kernel and balanced class weight  we get test accuracy of:  0.8
18346957311535

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3159         919
True >50k        181        1242

F1-score is: 0.6930803571428572
```

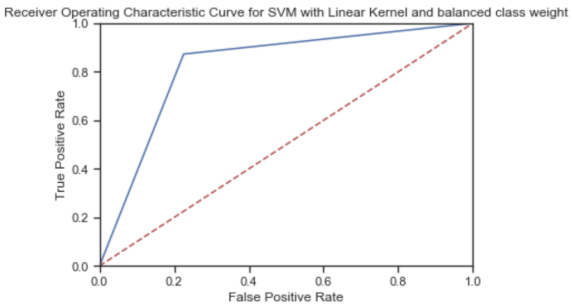

Figure 28: Results for the SVM with linear kernel with correction for class imbalance

```
---------- Results Summary for Dummy Classifier that predicts most frequent for the column: All Columns ----------

0.003930091857910156 seconds for  Dummy Classifier that predicts most frequent  to train
Using All Columns as a predictor for  Dummy Classifier that predicts most frequent  we get train accuracy of:  0.7425
805571967459
Using All Columns as a predictor for  Dummy Classifier that predicts most frequent  we get test accuracy of:  0.75804
39920014543

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       4170          0
True >50k       1331          0

F1-score is: 0.0
```

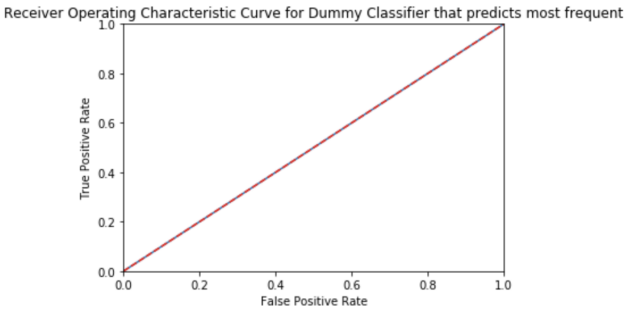

Figure 29: Results for the dummy classifier which always predicts the most frequent class

B. Further results which were not included in the final analysis

```
---------- Results Summary for SVM with Sigmoid Kernel for the column: All Columns ----------

28.699309825897217 seconds for  SVM with Sigmoid Kernel  to train
Using All Columns as a predictor for  SVM with Sigmoid Kernel  we get train accuracy of:  0.7793028223424079
Using All Columns as a predictor for  SVM with Sigmoid Kernel  we get test accuracy of:  0.8127272727272727

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3545        599
True >50k        605        752

F1-score is: 0.5553914327917282
```

Figure 30: Results for SVM with sigmoid kernel

```
---------- Results Summary for SVM with Radial Basis Function Kernel for the column: All Columns ----------

19.37322497367859 seconds for  SVM with Radial Basis Function Kernel  to train
Using All Columns as a predictor for  SVM with Radial Basis Function Kernel  we get train accuracy of:  0.85302004272
1447
Using All Columns as a predictor for  SVM with Radial Basis Function Kernel  we get test accuracy of:  0.845454545454
5455

Confusion Matrix:
           Pred <50k  Pred >50k
True <50k       3861        283
True >50k        562        795

F1-score is: 0.6529774127310062
```

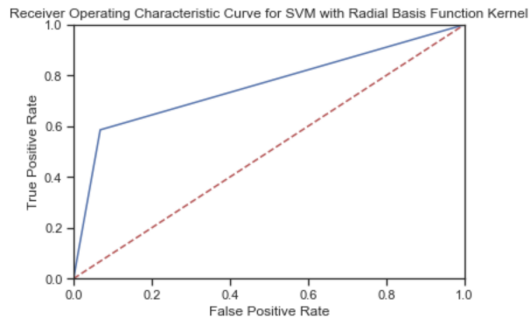

Figure 31: Results for SVM with radial basis function kernel