Part 1 - Kirsten Timmerman

Process

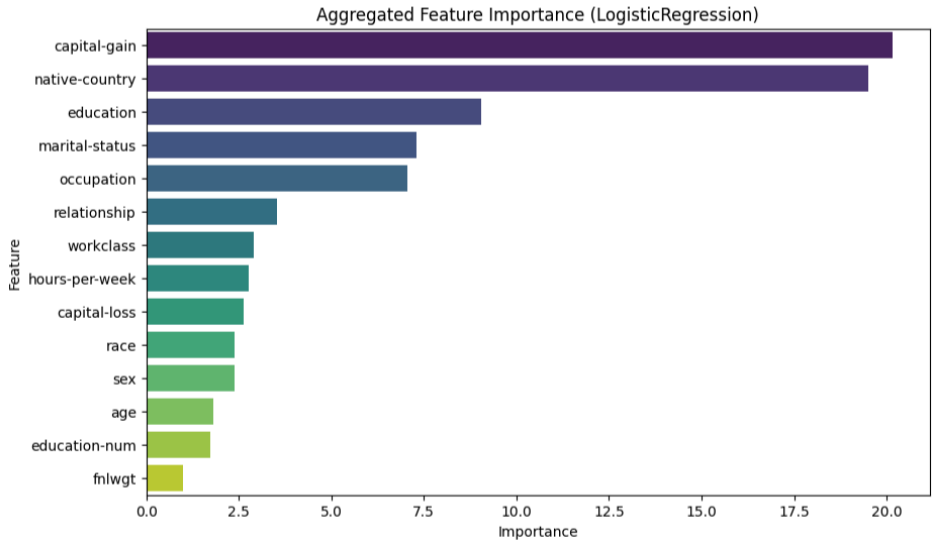
Preprocessing

First we preprocess the data.
We first consider missing features. We check the percentage of missing features of columns, and consider the rule of thumb: If the percentage of missing values is above 60%, remove the feature. By checking the percentage of missing features we observe that the maximum percentage of missing features is 5.75%. This means that we keep all features. Since all the features that contain missing elements are string values (and not numerical), we cannot numerically impute the values. So we remove all rows that have missing features, which is 3630 from the 48842 rows.
Furthermore, we turn string columns into categorical data and clean the ‘salary’ column, which contains duplicate values with different spelling - both “>50K” and “>50K.”. After this, the values of the target column are encoded as 0 or 1.
Then, all categorical features are one-hot encoded, to convert categories into binary indicators that models can interpret. Since there is no natural order to the categories, they can’t be turned into numerical values.
All numerical values are normalized, which is important as some models are sensitive to feature scaling.

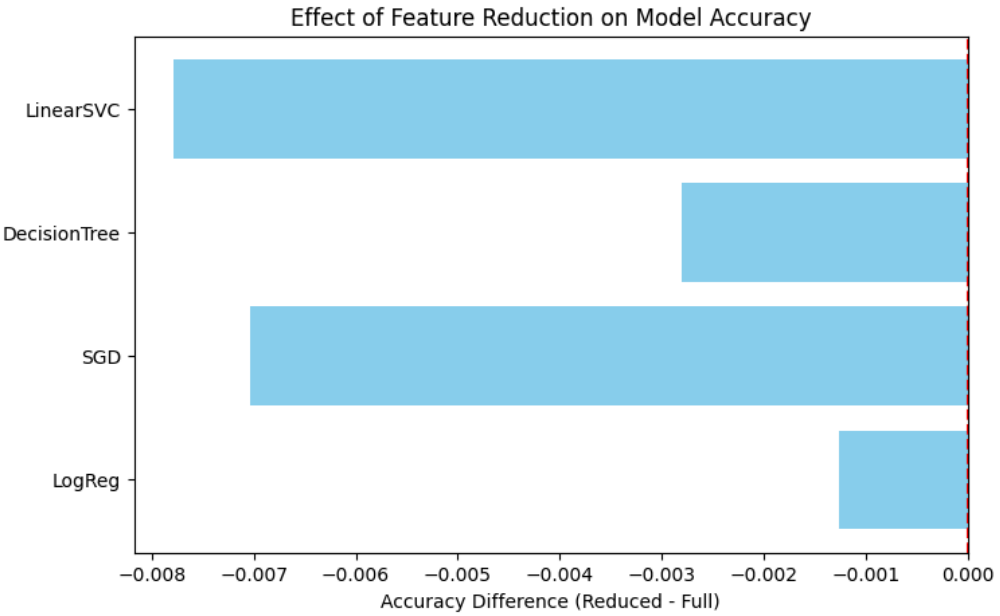
Feature selection & reduction

First we look at the domain and remove features that are irrelevant. Because the feature ‘education’ contains categorical data on the level of education, and ‘education-num’ refers to the years of education, we decided to remove ‘education’ as it is thus a redundant feature because they refer to the same data. Next to that, we removed the feature ‘fnlwgt’ as it does not contain any information on the actual person, but rather data that is used by the census for administrative purposes.

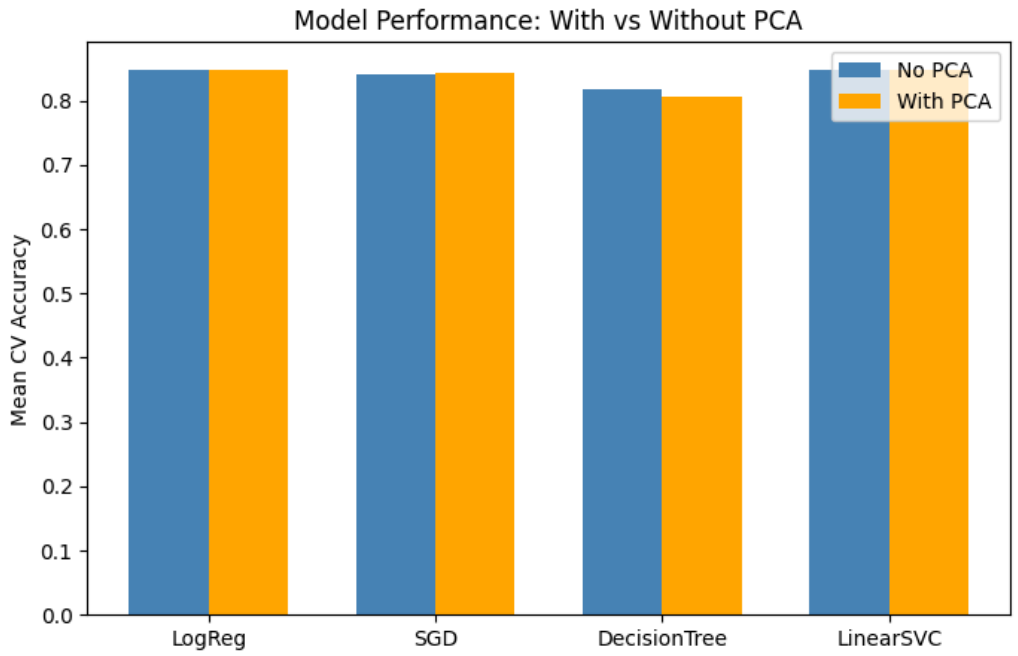
To preserve the interpretability of the results of the model, we initially decided to not apply dimensionality reduction, because the results are less interpretable and it’s more difficult to calculate feature importance. Rather, we would apply feature selection instead of feature importance. This means that per feature, the feature importance is calculated. An example of this can be seen in the figure below.



Looking at the feature importance chart, we observe that the last 3~5 features have a relatively low performance. Then, we set an absolute threshold of 1 to remove the features that have a relatively low feature importance. However, we noticed that both when setting an absolute threshold or a relative threshold (selecting the k features with the highest importance), the accuracy of the model was worse.



Because of this, we chose to apply PCA as dimensionality reduction, instead of feature selection. Since we one-hot encode the categorical values, we expanded the dataset up to 86 features. When we applied PCA of 60, we saw that this led to either the same performance, or either slightly worse or slightly better performance.



PCA with 60 components

Evaluation

To evaluate the performance of the different models, we look at the classification report, which shows the main classification metrics: precision, recall, f1-score and support.

$$\text{precision} = \frac{\text{tp}}{\text{tp}+\text{fp}}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp}+\text{fn}}$$

$$\text{accuracy} = \frac{\text{tp}+\text{tn}}{\text{tp}+\text{tn}+\text{fp}+\text{fn}}$$

$$\text{F}_1\text{ score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision}+\text{recall}}$$

	Logistic Regression	SGD Classifier	Decision Tree	Linear SVC
Precision	0.84	0.84	0.80	0.84
Recall	0.85	0.84	0.81	0.85
Accuracy	0.85	0.84	0.81	0.85
F1 score	0.84	0.84	0.80	0.84

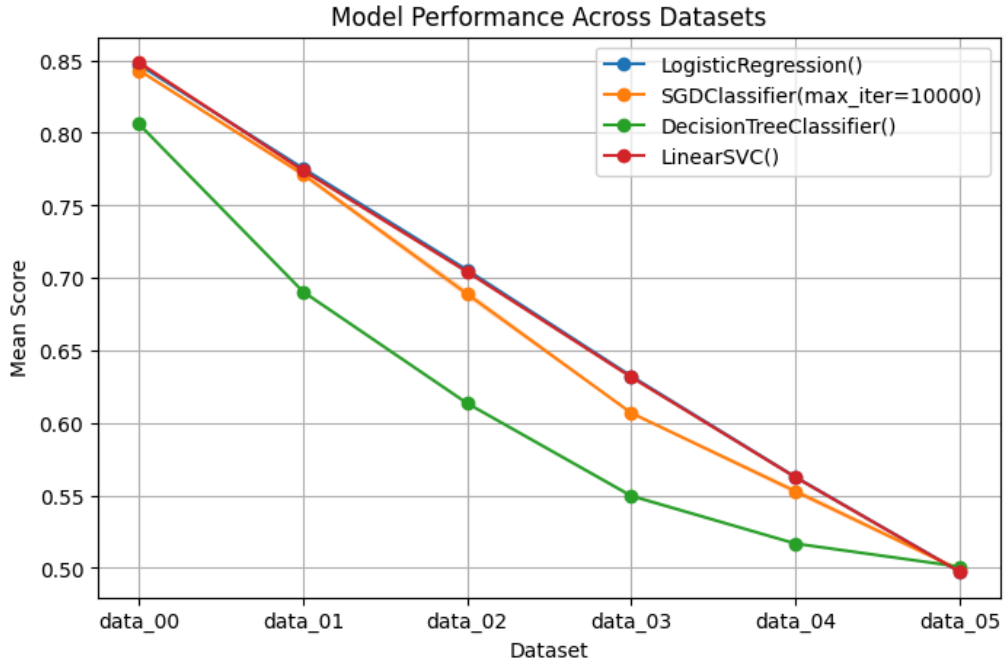
When we look at the actual results, we can see that Logistic Regression and Linear SVC perform the best.

Label Perturbation

The figure below shows the accuracy of each model across the different perturbed datasets.The variance is also plotted as a coloured area around the plotted line, however the variance is very small so no coloured area can be seen. We can partly explain this lack of variance by the fact that in our code we already apply cross-validation, meaning that per iteration we already calculate the score 5 times (n_splits=5) and take the mean. Then repeating the experiments 5 times again will not show that much variance.

For all of the models we can see a significant decrease in the accuracy. We can see that for LogisticRegression and for LinearSVC, that there is a linear decrease in accuracy when linearly decreasing the number of correctly labeled target values. We can see that the SGDClassifier performs slightly worse than the other two, and that the DecisionTreeClassifier performs the worst. For the performance of the decision tree, the accuracy of the classifier drops immediately once the data is perturbed and the noise is added.

We can explain this behaviour by the fact that decision trees are more sensitive towards noisy data and therefore the flipped labels. One flipped label can already cause a completely different tree structure. This means that decision trees are more likely to overfit and not generalize on the training data. The other models which are linear are able to generalize well and have less noise labels. That for data_05 (50% of the data is perturbed), the accuracy is around 50% for all models, makes sense as at that point it's just random guessing in binary classification, so on average 50% of the guesses are correct.



Discussion

First of all, it is important to look at the domain and try to find outliers or instances that do not make sense. In that case for example, it could also be possible to find domain experts which can look over the datasets and identify mislabeled points. However, when that is not possible (which is the case in our domain), it is essential to find a model that does not overfit and generalizes

well. This means that even in datasets with a lot of noise, it is still possible to find a general trend. If we find a dataset that we suspect of being very noisy or containing a lot of wrong labels, we need to find the trade-off between bias and variance. In this case, a low variance is important to reduce overfitting, and thus we might have to accept a high bias.
Besides choosing models that are less prone to overfitting, we might introduce a regularization parameter. This is also the case with the LogisticRegression model we used. This model adds an L2 penalty term, which penalizes too complex models. It is also possible to combine several models to ensure the influence of mislabeled instances. This can be ensemble methods like bagging or boosting.