Task 4 - Data Science for Engineers

We have done great progress in class regarding how and when to use different machine learning algorithms. Let us see a bit of what we have learnt up to this point:

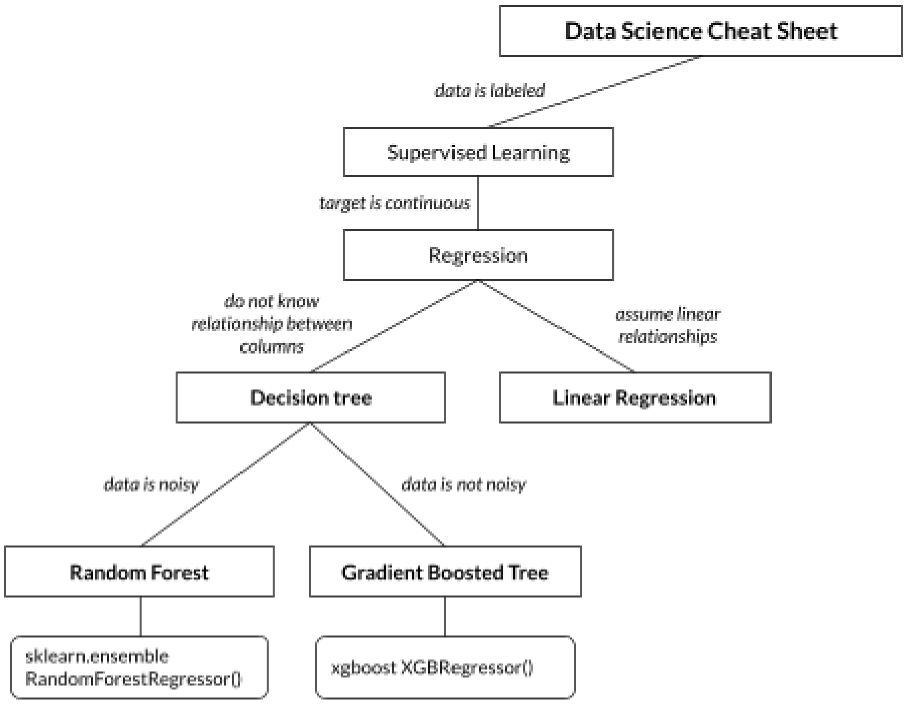


Image 1. Data Science Cheat Sheet

There is a key, common factor that all of these models hold, which actually stops us from using any of the above models for the present problem: our target must be continuous. So for this problem, where we are trying to predict basically a yes/no, we cannot use these models. Sure they are great for continuous data such as house prices or tips, but not for a binary output. The nature of any of the models is not suitable for what we want. Let us dive into other options.

If we look closely, all of those models have another characteristic in common: the word regressor. Going back to task 3, we saw there are both classifiers and regressors on sklearn, where a classifier works for predicting a label (such as yes/no), which means, mapping the class within which a target variable would most likely fall (Brownlee, J., 2017), while regression works for predicting a continuous variable, such as price. So there it is! We can use a classifier model, now let us dive into which one should we use. Let us first see how our cheat sheet is looking.

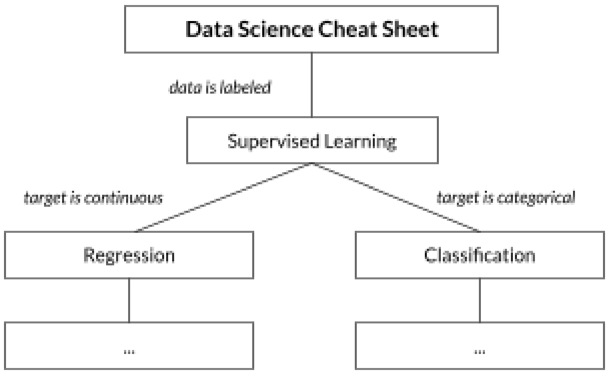


Image 2. Improved Cheat Sheet

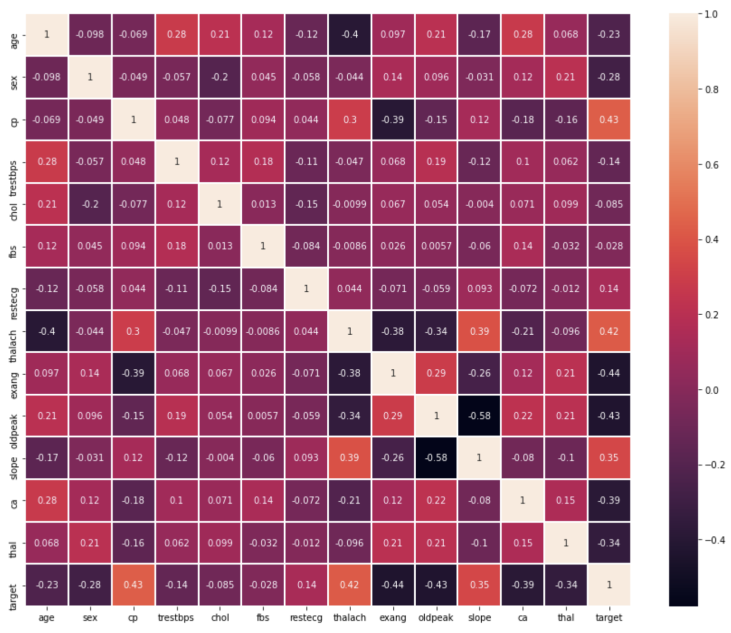
After some research, I decided to go on and develop four models and then choose the best, here are them with a brief description.

1. **Naive Bayes:** This algorithm assumes all features are independent, which means that they independently contribute to the probability of the target variable's class. This does not always happen, therefore the "Naive" part. Probability and likelihood values are calculated based on the frequency they appear in the data, and final probabilities are calculated using a formula called Bayes Theorem. (Ronaghan, S. 2018)
2. **K-Nearest Neighbors:** The K-nearest neighbors algorithm classifies objects depending on the nearest neighbor. The distance is measured using Euclidean distance. The core of the algorithm is to cluster data based on similarity amongst them, and when a new data point enters, we can classify it according to the distance (k) to a class. (Shah, D., Patel, S., Kumar, S., 2020). We can tweak the K parameter to look for better results.
3. **Decision Tree:** This model was previously seen. It basically tries to partition into "branches" according to the predictors and then make the prediction fall into a range or category. It is easy to use and can provide great results.
4. **Random Forest:** This is also a formerly studied model. What it does is develop multiple trees by bootstrapping rows and columns from the original data set to create *n* trees. When performing the prediction, the data is then analysed through the *n* trees and with a "majority-voting" policy, the final class is decided.
5. **XG Boost:** Finally, this is also an explored model. Gradient boosting is one of the most used algorithms out there. Especially for data that is not noisy. Here, a prediction model is produced in the form of an ensemble of weak prediction models, which commonly outperforms random forest.

It is also to note that classification techniques are good for supervised learning tasks with use cases as this specific one (predicting a clinical diagnosis based upon symptoms, laboratory results, and historical diagnosis). (Ronaghan, S. 2018)

Exploratory Data Analysis

As always, the goal of my EAD is to find relations between features to choose the best.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

Image 3. Correlation heat map

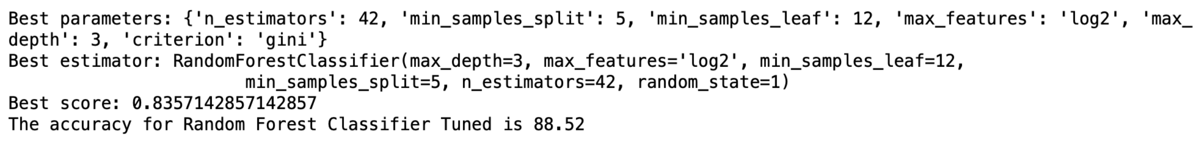
With this first correlation map I think we can drop two variables from the beginning: cholesterol serum and fasting blood sugar, which both have a very small correlation, almost zero. Let us remember that values close to +1 indicate the presence of a strong positive relationship between X and Y, while those close to -1 indicate a strong negative relationship between X and Y. Values close to zero mean that there is no relationship between X and Y. So in my model development I might drop those.

Model Development

There were three main takeaways for this task model development that I would like to point out for further reference.

1. Feature Engineering: This definitely was the most challenging part. On the last task at least I had some domain knowledge regarding home prices, such as how the number of rooms increments the price, that some areas (such as suburbs) have more capital gain than others and so on. But for this model I was completely blind. I am not a doctor and I have zero knowledge regarding how a set of factors can correlate to having a heart disease. While I did separate numerical and categorical variables and some models gave better scores by just using one of them, I did not feel confident enough to discard any variable. Taking this into account, we might perform better by developing ML models of areas we actually have some knowledge about.
2. Linear Correlation does not always suffice: Following up on the last issue, we might try to find relations between variables by doing some EAD, as I did. And while I did see that there was a small correlation between the target and cholesterol serum/fasting blood sugar, dropping then resulted in models with a lower accuracy score. Again, either a way deeper analysis, or domain knowledge will actually help us determine which variables to use. Not just a correlation table.

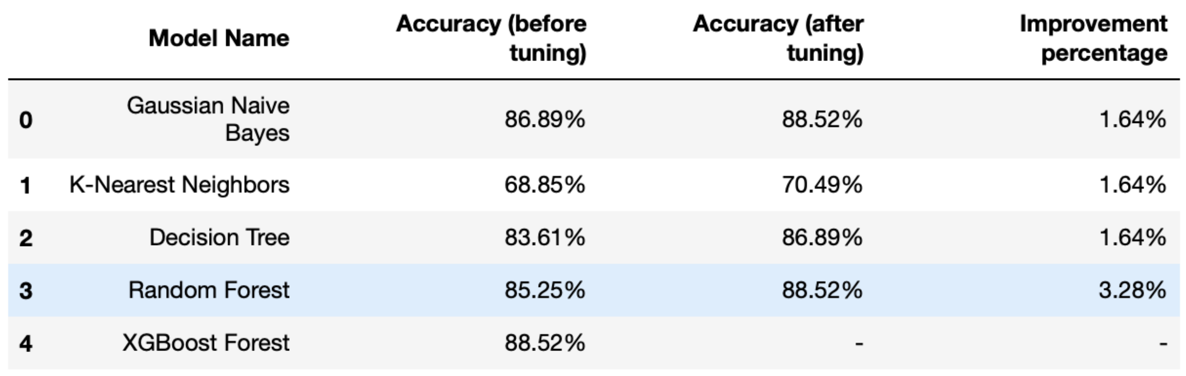
3. Hyper parameter tuning is a hard task: When you hear what hyper parameter tuning does, you might be convinced that it is easy to use, just choose which parameters you want to tweak and let sklearn do the hard part, but it is not true. Choosing the values to test the parameters with is difficult and providing many options to them is very CPU consuming. I even omitted trying to boost the XGBClassifier due to its large number of parameters and not that big domain of them. At the end I decided to dive deeper and try to improve even more the model I was most familiar with: Random Forest Classifier. The results were great and that was the model with the best accuracy. So in conclusion, sometimes it is better to hyper parameter tune one model rather than many.



```
Best parameters: {'n_estimators': 42, 'min_samples_split': 5, 'min_samples_leaf': 12, 'max_features': 'log2', 'max_
depth': 3, 'criterion': 'gini'}
Best estimator: RandomForestClassifier(max_depth=3, max_features='log2', min_samples_leaf=12,
                       min_samples_split=5, n_estimators=42, random_state=1)
Best score: 0.8357142857142857
The accuracy for Random Forest Classifier Tuned is 88.52
```

Image 4. Hyper parameter tuned Random Forest Classifier

According to the five models accuracy score, the best models are either the Gaussian Naive Bayes or the Tuned Random Forest. Below you can see a dataframe with the summary of the results.



	Model Name	Accuracy (before tuning)	Accuracy (after tuning)	Improvement percentage
0	Gaussian Naive Bayes	86.89%	88.52%	1.64%
1	K-Nearest Neighbors	68.85%	70.49%	1.64%
2	Decision Tree	83.61%	86.89%	1.64%
3	Random Forest	85.25%	88.52%	3.28%
4	XGBoost Forest	88.52%	-	-

Image 5. Accuracy results

There is one final thing I would like to do in order to check if my chosen model, the Random Forest Classifier, performed good predictions. For this, I used a confusion matrix, which is great to check the precision of a model with a binary target. Let us recall basic terminology and the significance of each cell of the matrix:

- True Positives (TP): Bottom right cell. The classifier **correctly** predicted that they **do** have the disease.
- True Negatives (TN): Upper left cell. The classifier **correctly** predicted they **don't** have the disease.
- False Positives (FP): Upper right cell. The classifier **incorrectly** predicted that they **do** have the disease, when in fact, they **don't.**
- False Negatives (FN): Bottom left cell. The classifier **incorrectly** predicted that they **do not** have the disease, when in fact, they **do.**

n = 61	**Predicted: 0**	**Predicted: 1**
Actual: 0	25	4
Actual: 1	4	28

Image 6. Confusion Matrix results for Random Forest Classifier

Above can be seen the results for the confusion matrix of one of our best models (according to its accuracy score), the Random Forest Classifier. The results are favorable and can be read as follows: From our total 61 test cases we predicted 28 cases correctly where the patient has the disease (and just 4 incorrectly); on the other hand, we predicted 25 cases also correctly where the patient does not have the disease (and just 4 incorrectly). Great results!

As a final note, next task I will try to stick with one or two models at most, and try to improve them as much as I can. Hyper parameter tuning is a very time consuming task (for both the CPU and me) and I think I can achieve better results if I stick with a model and tweak its parameters, as I did with the Random Forest Model.

References

1. D. Shah, S. Patel, and S. K. Bharti, “Heart Disease Prediction using Machine Learning Techniques,” *SN Computer Science*, vol. 1, no. 6, 2020.
2. J. Brownlee, “Difference Between Classification and Regression in Machine Learning,” Machine Learning Mastery, 21-May-2019. [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/. [Accessed: 08-Feb-2021].
3. S. Ronaghan, “Machine Learning: Trying to classify your data,” Medium, 07-Aug-2018. [Online]. Available: https://srnghn.medium.com/machine-learning-trying-to-predict-a-categorical-outcome-6ba542b854f5. [Accessed: 14-Feb-2021].