Assignment 1 – Classification using Scikit-learn

Student Name: <Ibrahim Alhalaki> **Student ID**: <25100728> **Programme**: <4BCT>

Algorithm 1 – Support Vector Regression

Support Vector Regression is the regression model that comes from the same group as Support Vector Machines. Support Vector Regression attempts to predict continuous values, which allows it to be suitable for tasks such as estimating tensile strength in steel. The main point of SVR is to find a function that fits the data, all while keeping the errors within a margin. SVR doesn't try to perfectly fit every single data point; instead, it focuses on finding a smooth curve that captures the relationship between the input features and the targets you want to predict (Vapnik, 1995; Smola & Schölkopf, 2004). Hence, SVR tends to be less sensitive to outliers and more robust compared to basic regression approaches. In scikit-learn, SVR is usually used with kernel functions, such as the Radial Basis Function, which allows SVR to model complex nonlinear patterns in the data (Pedregosa eta., 2011).

Detailed Description of Algorithm 1.

SVR creates a tolerance margin where small errors are ignored around the regression line. Predictions that fall inside this margin are treated as a pass; this means the model will not spend resources trying to fix or correct them. This is different from classical regression, which penalises every deviation, no matter how small or invaluable it is (Smola & Schölkopf, 2004).

To control how strict SVR should be, it uses a parameter called **C**, which determines how much the model should punish errors. A high C value means the model will try harder to match the training data, which in turn risks overfitting, while a lower C value allows the model to be more lenient and generalise better (Vapnik, 1995).

Since numerous real-world relationships between inputs and outputs are non-linear, SVR uses kernel functions to learn curved patterns. In this assignment

(need to finish this)

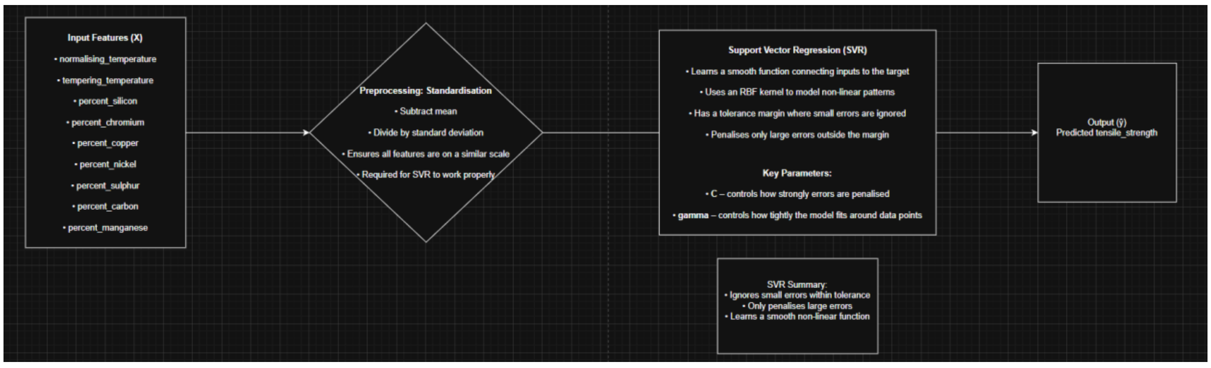

Input Features (X)
• normalising_temperature
• tempering_temperature
• percent_silicon
• percent_chromium
• percent_copper
• percent_nickel
• percent_sulphur
• percent_carbon
• percent_manganese
Preprocessing: Standardisation
• Subtract mean
• Divide by standard deviation
• Ensures all features are on a similar scale
• Required for SVR to work properly
Support Vector Regression (SVR)
• Learns a smooth function connecting inputs to the target
• Uses an RBF kernel to model non-linear patterns
• Has a tolerance margin where small errors are ignored
• Penalises only large errors outside the margin
Key Parameters:
• C – controls how strongly errors are penalised
• gamma – controls how tightly the model fits around data points
Output (ŷ)
Predicted tensile_strength
SVR Summary:
• Ignores small errors within tolerance
• Only penalises large errors
• Learns a smooth non-linear function

Figure 1: How the SVR model works

Why I chose this algorithm.

I chose SVR because of how effective it is when it comes to non-linear relationships between the inputs and the target, and that is not captured by simple regression models. The model is also more robust when it comes to small fluctuations in data since it doesn't take into account the minor errors and instead focuses on learning the overall trend. This makes SVR a good fit for the dataset provided to us in this project. **(Need to finish this)**

Hyperparameter Details for Tuning.

Hyperparameter 1: **C**

C controls how strict the model punishes large errors. When running with a high value of C, the model tries harder to fit the training data, while a low value of C keeps the model simple. Tuning C helps balance overfitting and underfitting.

Hyperparamer 2: Gamma

Gamma controls how much influence each training model point is allowed on the model. High values of Gamma make the model very sensitive to small patterns, while having low values of gamma makes it smoother.

Algorithm 2 – Random Forest Regressor

Random Forest Regression is a learning algorithm that builds several decision trees and combines their predictions to form one result. Random forest regressor does not rely on one single tree; Random forest averages the results of the different trees to create a more stable, better prediction (Breiman, 2001). The trees in the algorithm are trained on different subsets of the data we provide; only a random subset of features is considered at each split, which allows us to reduce correlation between trees and overall performance.

Detailed Description of Algorithm 2.

The model works by using something called bootstrap aggregation. This means that every tree in the forest is trained on a sample of the dataset taken with replacement, which means that every tree sees a different version of the data (Breiman, 2001). Since the trees are independently trained on different samples, each tree learn different patterns and error tendencies. When all the trees are combined by averaging their predictions, the algorithm becomes less sensitive to noise and much more stable.

Another feature of Random Forest is random feature selection. At each split in a tree, the algorithm examines a random subset of the available features (Ho, 1995). That ensures that the trees in the forest do not all make the same decisions, which improves the diversity and prevents overfitting.

Random Forest also captures the non-linear relationships and the feature interactions without needing any special scaling or transformations. An example of that is, if we have two features that only matter when combined (like composition + temperature), a Random forest can't detect that relationship automatically.

One of the useful outputs of Random Forest is feature importance, the feature importance measures how much each feature matters for prediction. This is helpful because It can highlight which measurements are most influential in finding the target value.

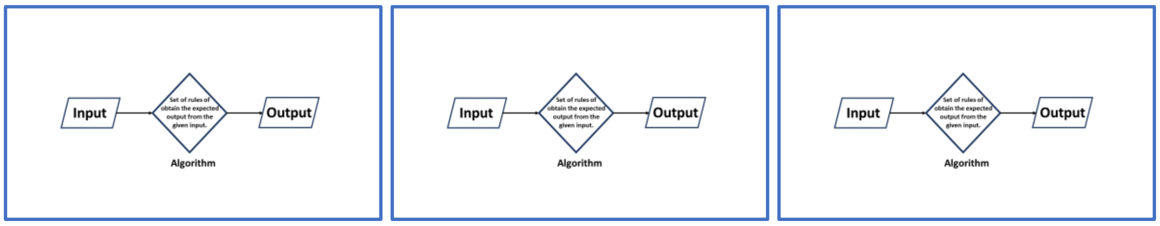

Input
Set of rules of obtain the expected output from the given input.
Output
Algorithm
Input
Set of rules of obtain the expected output from the given input.
Output
Algorithm
Input
Set of rules of obtain the expected output from the given input.
Output
Algorithm

Figure 2: Graphic illustration of how the algorithm works

Why I chose this algorithm.

I chose this algorithm because it is one of the most reliable and widely used algorithms for structured datasets. Unlike SVR, random forest doesn't need scaling, and that makes it naturally handle a wide range of input values, which makes it a good fit for the dataset provided.

Hyperparameter Details for Tuning.

Hyperparameter 1: n_estimators

This hyperparameter controls how many trees are included in the forest. Usually, using more trees improves the accuracy and stability because the final prediction is based on a larger ensemble, which in turn lowers variance and the risk of overfitting. But the downside for that is, it raises the computational power needed since more trees need to be trained and stored(Breiman,2001; Scikit learn,2024)

Hyperparameter 2: max_depth

This parameter limits how much depth each tree in the forest can grow. Trees with more depth can capture more complex patterns, but if the depth is too large, the model may overfit the training data. Putting a restriction on the depth helps make the trees simpler, improving generalisation to unseen data. Balancing the depth of the trees is therefore important: shallow trees may underfit, while overly deep trees may overfit (DataCamp, 2024; Scikit-learn, 2024).

Algorithm 1 - <Name of Algorithm> - Model Training and Evaluation

To do

Data Preprocessing and Visualisation

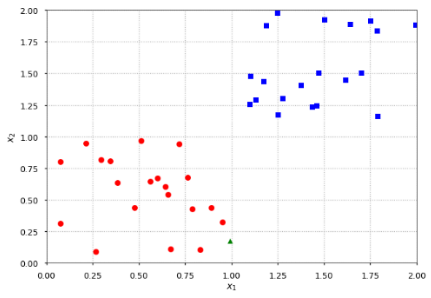
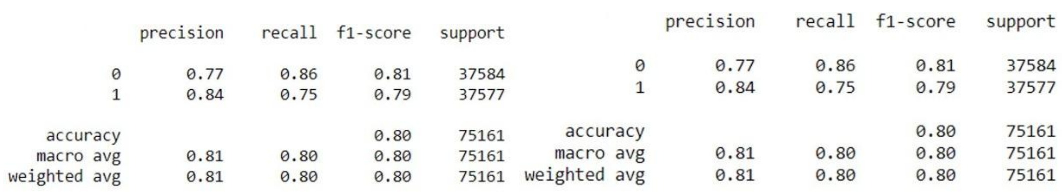


Figure 3: Visualisation of the dataset before training

Training and Evaluation Details

To do



```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81     37584
           1       0.84      0.75      0.79     37577

    accuracy                           0.80     75161
   macro avg       0.81      0.80      0.80     75161
weighted avg       0.81      0.80      0.80     75161
```

```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81     37584
           1       0.84      0.75      0.79     37577

    accuracy                           0.80     75161
   macro avg       0.81      0.80      0.80     75161
weighted avg       0.81      0.80      0.80     75161
```

Figure 4: Summary of Results Achieved from Training and Testing

Discussion of results

To do

Algorithm 2 - <Name of Algorithm> - Model Training and Evaluation

To do

Data Preprocessing and Visualisation

To do

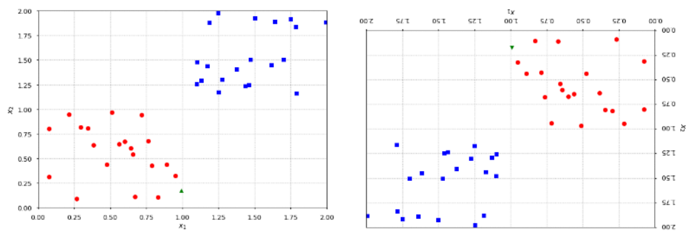
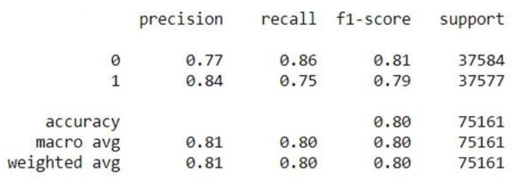


Figure 5: Visualisation of the dataset before and after pre-processing was applied

To do

Training and Evaluation Details

To do



```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81     37584
           1       0.84      0.75      0.79     37577

    accuracy                           0.80     75161
   macro avg       0.81      0.80      0.80     75161
weighted avg       0.81      0.80      0.80     75161
```

Figure 6: Summary of Results Achieved from Training and Testing

To do

Discussion of results

To do.

Conclusions

To do

Key Findings

- To do

Comparative Analysis of Algorithm Performances

To do

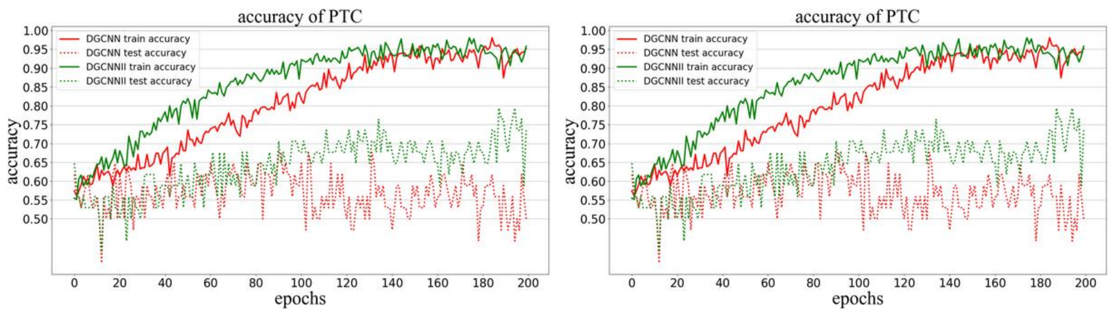


Figure 7: Graphed comparison of results from the two algorithms

Recommended Hyperparameter Valued based on Results

To do

Concluding Remarks

To do

References

1. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144–152). ACM.
2. Cherkassky, V., & Ma, Y. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1), 113–126. https://doi.org/10.1016/S0893-6080(03)00169-2
3. Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In *Advances in Neural Information Processing Systems* (pp. 155–161).Reference any books, articles, blogs, documentation, videos etc. here…
4. Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). A practical guide to support vector classification. Technical Report. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdfReference any books, articles, blogs, documentation, videos etc. here…
5. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.Reference any books, articles, blogs, documentation, videos etc. here…
6. Scikit-learn Developers. (2024). *sklearn.svm.SVR documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.htmlReference any books, articles, blogs, documentation, videos etc. here…
7. Scikit-learn Developers. (2024). *Guide to hyperparameter tuning with GridSearchCV*. https://scikit-learn.org/stable/modules/grid_search.htmlReference any books, articles, blogs, documentation, videos etc. here…

Report Update Log

Date	Update Description	Time Spent
11/10/25	Researched the different types of regression models	3 hours
11/11/25	Picked the models and started learning about them	30 min
11/12/25	Began with collecting resources and saving them.	1 hour
11/13/25	Started with writing the paper, starting with the introduction to SVR	1:30 Hours
11/14/25	Started with the second model	1 hour
11/15/25	Began experimenting with	3 hours
11/16/25	Completed the halfway point	6 hours
11/17/25		2 hours
11/18/25		4 hours
11/19/25		30 mins
11/20/25		1 hour
11/21/25		3 hours
11/22/25		30 mins
11/22/25		1 hour
11/23/25		
11/24/25		
11/25/25		
11/26/25		
11/27/25		

To do list:
make a diagram representing how random forest regressor works.