SIT720 Machine Learning: Final Machine Learning Project

SIT720 - Machine learning

Student ID: 222355509

Name: Mark Hashimoto

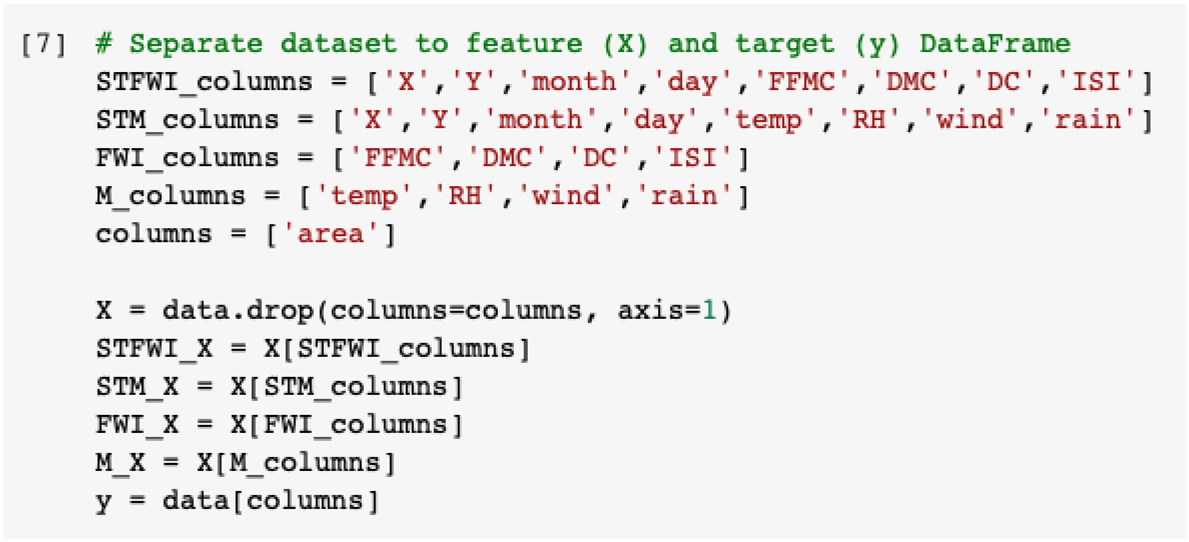
1. Read the article and reproduce the RMSE results presented in Table 3 using Python modules and packages (including your own script or customised codes).

i) you should use the same set of features used by the authors.

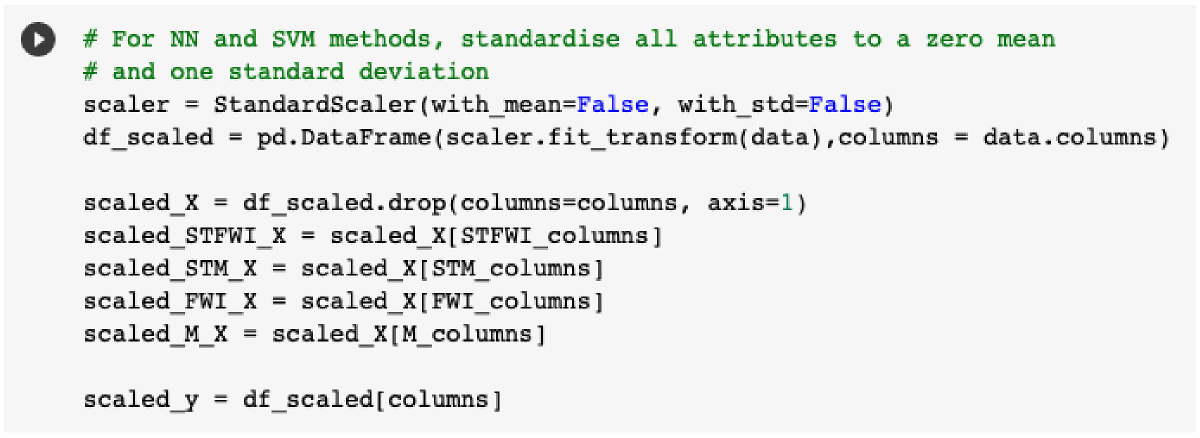
Cortez and Morais [1] describe sets of features as follows:

- STFWI – using spatial (‘X’, ‘Y’), temporal (‘month’, ‘day’) and the four FWI components (‘FFMC’, ‘DMC’, ‘DC’, ‘ISI’)
- STM – with the spatial (‘X’, ‘Y’), temporal (‘month’, ‘day’) and four weather variables (‘temp’, ‘RH’, ‘wind’, ‘rain’)
- FWI – only the four FWI components (‘FFMC’, ‘DMC’, ‘DC’, ‘ISI’)
- M – four weather conditions (‘temp’, ‘RH’, ‘wind’, ‘rain’)

[1] also modify all attributes standardised to a zero mean and one standard deviation for the Neural networks and Support vector machines methods.The snippet of code setting those features and standardisation are as follows:



```
[7] # Separate dataset to feature (X) and target (y) DataFrame
    STFWI_columns = ['X','Y','month','day','FFMC','DMC','DC','ISI']
    STM_columns = ['X','Y','month','day','temp','RH','wind','rain']
    FWI_columns = ['FFMC','DMC','DC','ISI']
    M_columns = ['temp','RH','wind','rain']
    columns = ['area']

    X = data.drop(columns=columns, axis=1)
    STFWI_X = X[STFWI_columns]
    STM_X = X[STM_columns]
    FWI_X = X[FWI_columns]
    M_X = X[M_columns]
    y = data[columns]
```

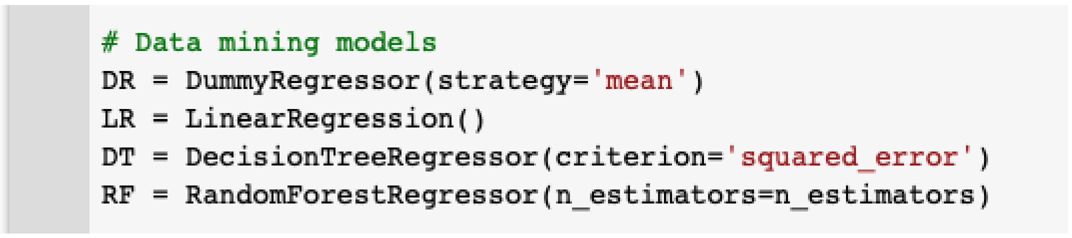


```python
# For NN and SVM methods, standardise all attributes to a zero mean
# and one standard deviation
scaler = StandardScaler(with_mean=False, with_std=False)
df_scaled = pd.DataFrame(scaler.fit_transform(data),columns = data.columns)

scaled_X = df_scaled.drop(columns=columns, axis=1)
scaled_STFWI_X = scaled_X[STFWI_columns]
scaled_STM_X = scaled_X[STM_columns]
scaled_FWI_X = scaled_X[FWI_columns]
scaled_M_X = scaled_X[M_columns]

scaled_y = df_scaled[columns]
```

ii) you should use the same classifier with exact parameter values.

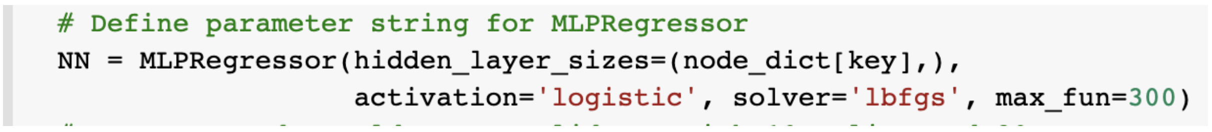
As described by [1], following models with parameter values are used:

- Naive average predictor: In order to predict the mean of the training dataset, DummyRegressor class with parameter strategy: ‘mean’ is used.
- Multiple regression: LinearRegression class is used to predict a dependent value from several independent variables.
- Decision tree: DecisionTreeRegressor class with parameter criterion: ’squared_error’ is used to satisfy “DT node split was adjusted for the reduction of the sum of squares”.
- Random forest: RandomForestRegressor with parameter n_estimators: 500 is used to meet “the default parameters were adopted for the RF (e.g. T = 500)”



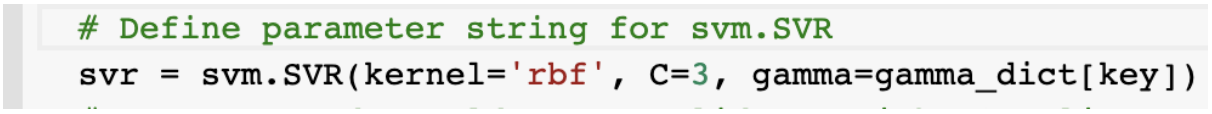
```python
# Data mining models
DR = DummyRegressor(strategy='mean')
LR = LinearRegression()
DT = DecisionTreeRegressor(criterion='squared_error')
RF = RandomForestRegressor(n_estimators=n_estimators)
```

- Neural network: In order to replicate the model performed by [1], “the NN were adjusted using NR = 3 trainings and E = 100 epochs of the BFGS algorithm”, MLPRegressor class with parameters hidden_layer_sizes:(4,), activation:'logistic', solver:'lbfgs', max_fun:300 is used.



```python
# Define parameter string for MLPRegressor
NN = MLPRegressor(hidden_layer_sizes=(node_dict[key],),
                  activation='logistic', solver='lbfgs', max_fun=300)
```

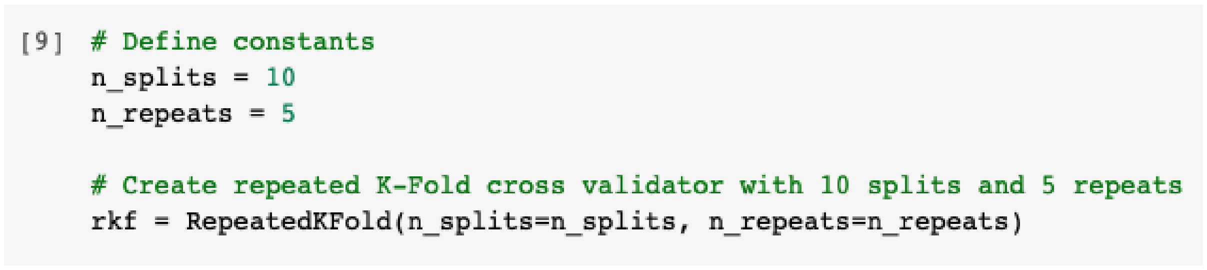
- Support vector machine: svm.SVR class is used to perform “Sequential Minimal Optimization algorithm was used to fit the SVM”.



```python
# Define parameter string for svm.SVR
svr = svm.SVR(kernel='rbf', C=3, gamma=gamma_dict[key])
```

iii) you should use the same training/test splitting approach as used by the authors.

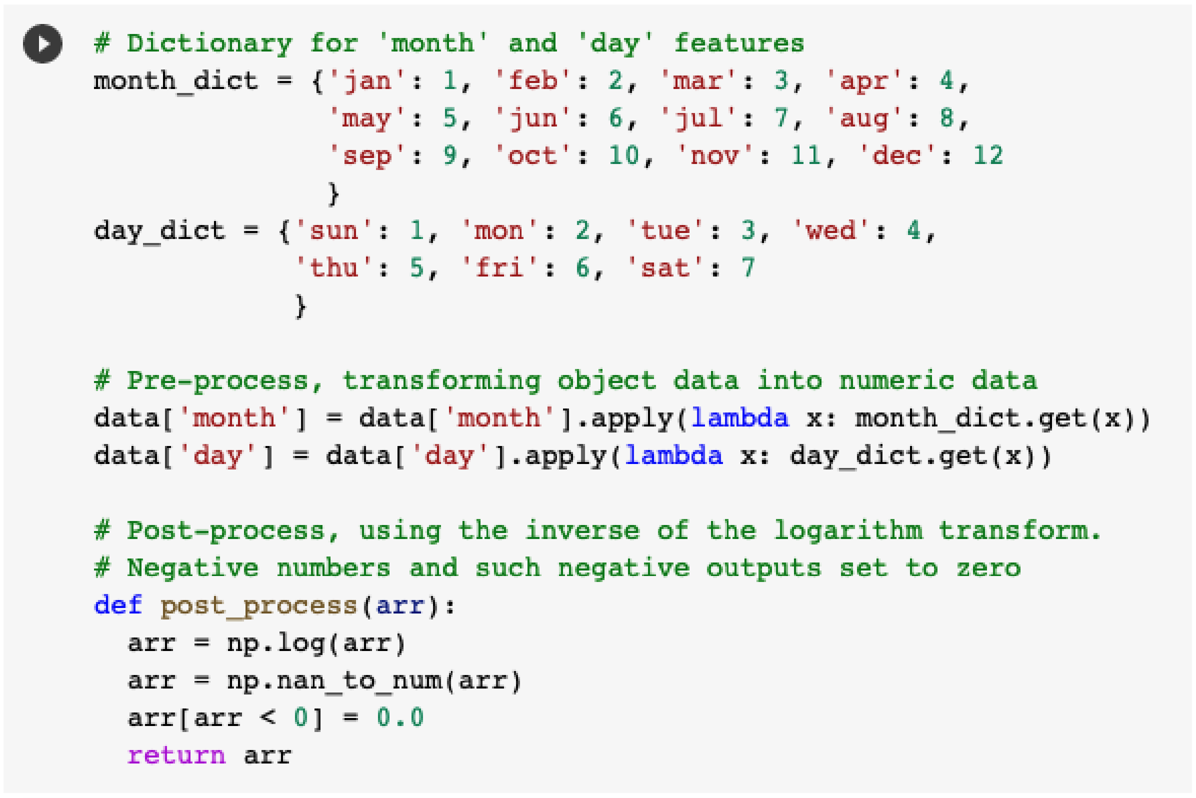
As described in [1], “To access the predictive performances, thirty runs of a 10-fold [17] (in a total of 300 simulations) were applied to each tested configuration.”, RepeatedKFold class is used to replicate the splitting approach. Due to the time constraints for processing 300 simulations, this task applies five runs of a 10-fold (in a total of 50 simulations) with parameters n_splits: 10, n_repeats: 5.



```
[9]  # Define constants
     n_splits = 10
     n_repeats = 5

     # Create repeated K-Fold cross validator with 10 splits and 5 repeats
     rkf = RepeatedKFold(n_splits=n_splits, n_repeats=n_repeats)
```

iv) you should use the same pre/post processing, if any, used by the authors.

For data pre-processing, [1] explains “The nominal variables (i.e. discrete with more than two non-ordered val- ues), such as the month and day, were transformed into a 1-of-C encoding”. In terms of data post-processing, [1] performed “After fitting the DM models, the outputs were postprocessed using the inverse of the logarithm transform. In few cases, this transformation may lead to negative numbers and such negative outputs were set to zero”. The snippet code to replicate the pre/post processing is as follows:



```
# Dictionary for 'month' and 'day' features
month_dict = {'jan': 1, 'feb': 2, 'mar': 3, 'apr': 4,
              'may': 5, 'jun': 6, 'jul': 7, 'aug': 8,
              'sep': 9, 'oct': 10, 'nov': 11, 'dec': 12
              }
day_dict = {'sun': 1, 'mon': 2, 'tue': 3, 'wed': 4,
            'thu': 5, 'fri': 6, 'sat': 7
            }

# Pre-process, transforming object data into numeric data
data['month'] = data['month'].apply(lambda x: month_dict.get(x))
data['day'] = data['day'].apply(lambda x: day_dict.get(x))

# Post-process, using the inverse of the logarithm transform.
# Negative numbers and such negative outputs set to zero
def post_process(arr):
  arr = np.log(arr)
  arr = np.nan_to_num(arr)
  arr[arr < 0] = 0.0
  return arr
```

v) you should report the same performance metric (RMSE) as shown in Table 3.

The table below is the summary of performance metric (RMSE) in each method and dataset. Overall, RMSE scores are better than the ones in the literature, around 46.0 in this report whereas around 64.0 in the literature. However, the score differences in each dataset are minimal (the maximum difference is 1.976 in STFWI dataset), hence, the experiment protocols are performed well.

	STFWI	STM	FWI	M
Naive	47.400	46.501	45.244	**45.962**
Multiple regression	46.196	46.673	45.554	46.350
Decision tree	46.869	47.196	45.761	47.230
Random forest	46.468	**46.122**	**<u>45.234</u>**	47.499
Neural networks	46.153	47.103	47.192	47.238
Support Vector machines	**45.424**	46.312	47.105	46.788

2. Design and develop your own ML solution for this problem.

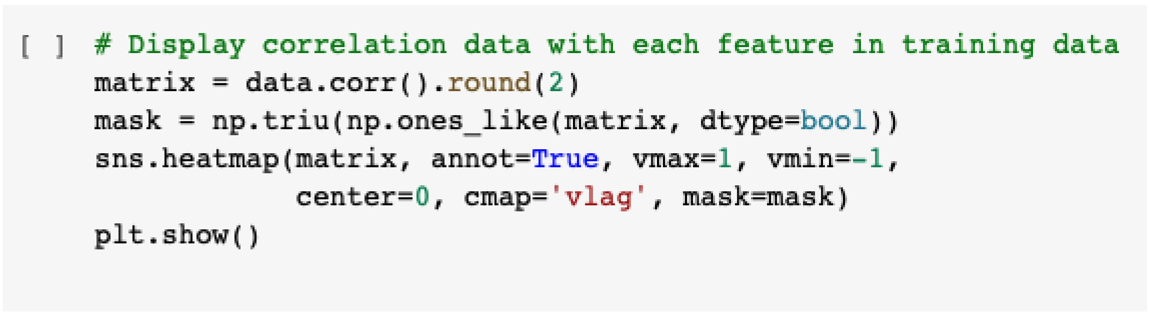
i) Motivation behind the proposed solution.

This task explores following three approaches to propose possible solutions:

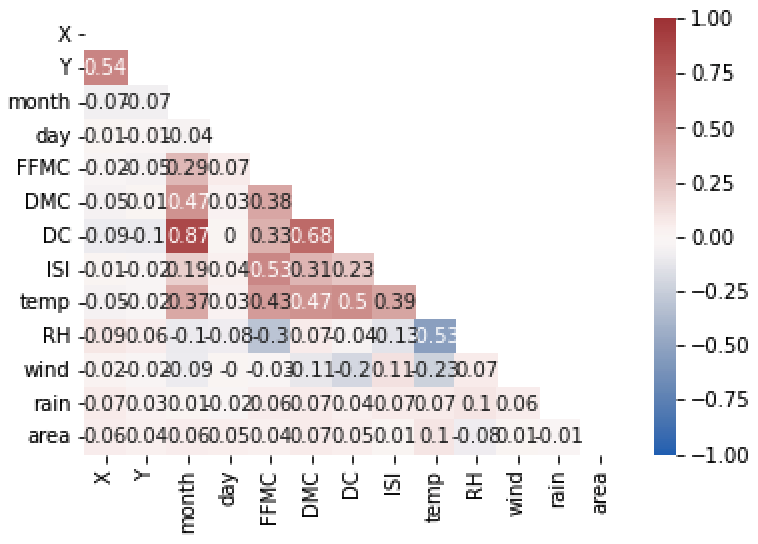
1. Dataset selection based on correlation analysis and importance score

Correlation analysis reveals how dependent each independent variable is in relation to the target variable (‘area’). The results show that there is a weak correlation between target variable (‘area’) and other independent variables, which concludes that this analysis does not provide much insight into the dataset.

Correlation analysis



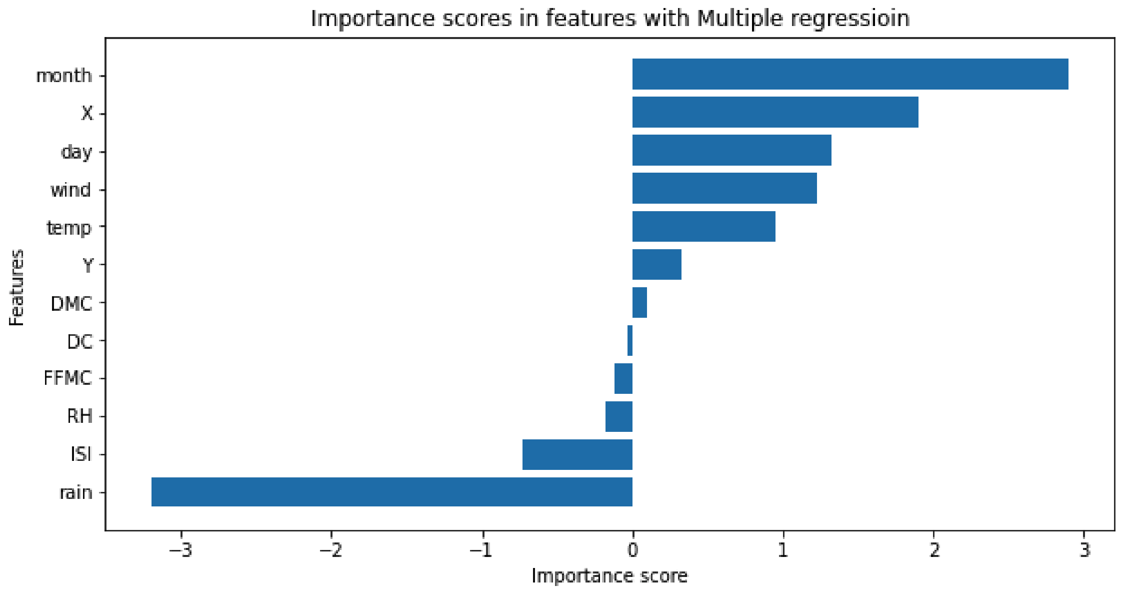
```
[ ] # Display correlation data with each feature in training data
    matrix = data.corr().round(2)
    mask = np.triu(np.ones_like(matrix, dtype=bool))
    sns.heatmap(matrix, annot=True, vmax=1, vmin=-1,
                center=0, cmap='vlag', mask=mask)
    plt.show()
```



Importance score provides scores to input features with respect to predictive models, which suggests how important each independent variable is when making a prediction with each model. Even though there are twelve features in the dataset, which is not a large number, correlation analysis and importance scores provide insight into data and models to have a better understanding of the problem.
The results of important scores of multiple regression, decision tree, and random forest are as follows:

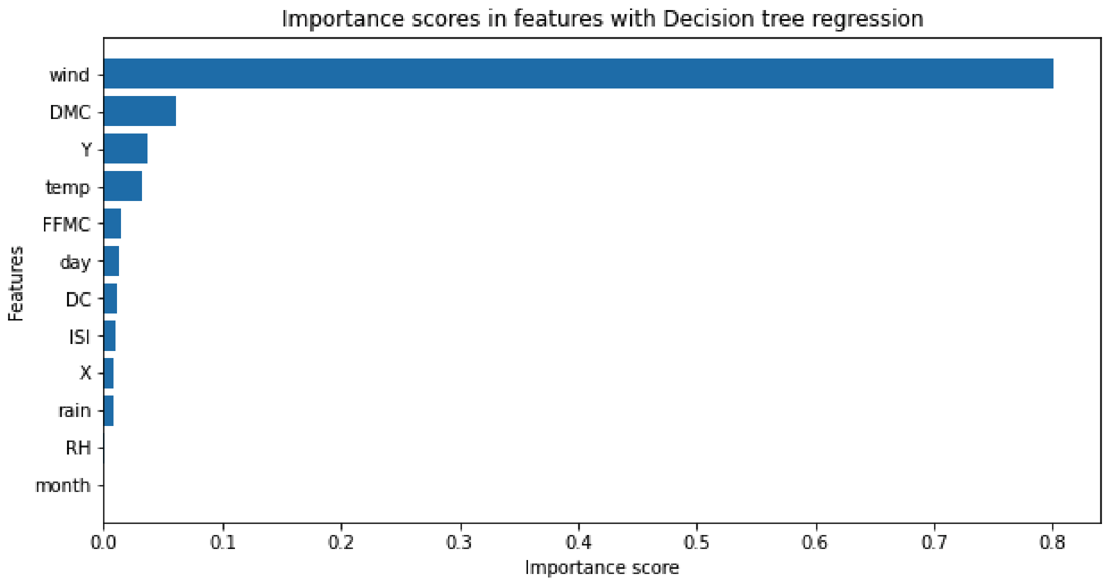
- Multiple regression

Seven features ('month', 'X', 'day', 'wind', 'temp', 'Y', and 'DMC') are important features with multiple regression.



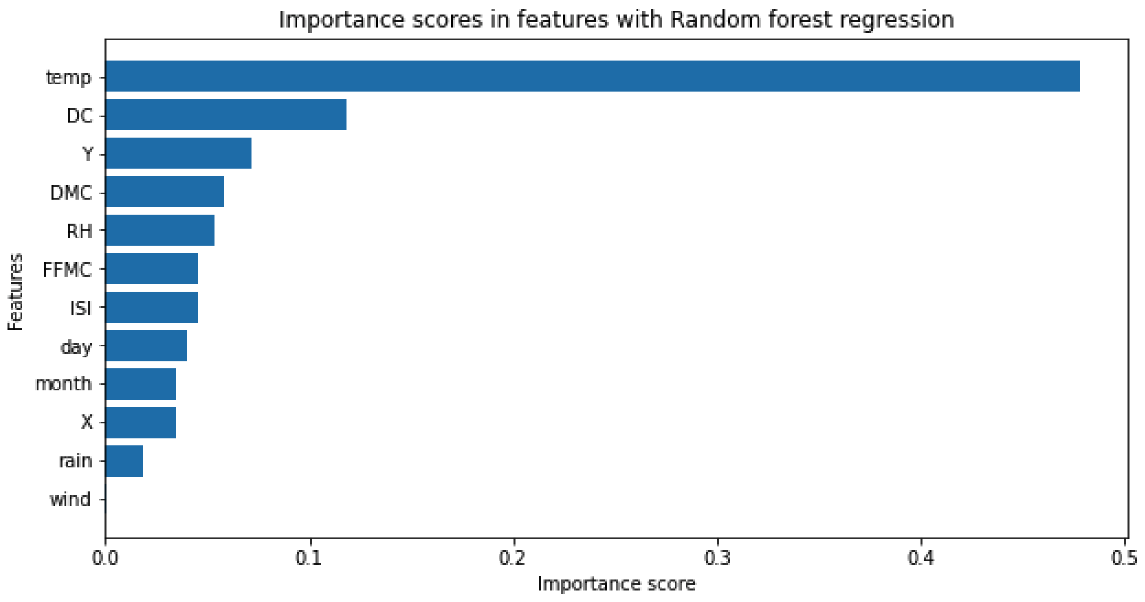
- Decision tree regression

Four features ('wind', 'DC', Y, and 'temp') are important features with decision tree regression



- Random forest

One feature (temp) has a strong relationship with random forest, and four features ('DC', 'Y', 'DMC', and 'RH') are following.



The table below is the summary of importance scores in each method. Although there are varied results in each method, eight features ('X', 'Y', 'day', 'month', 'wind', 'temp', 'DC', and 'DMC') are relatively important features amongst all techniques, therefore, new dataset with these features will be tested.

	Multiple regression	Decision tree	Random forest
1	month	wind	temp
2	X	DMC	DC
3	day	Y	Y
4	wind	temp	DMC
5	temp		RH
6	Y		
7	DMC		

2. Hyperparameter tuning

Since the models in [1] used default hyperparameters, there is still room to explore customising the aforementioned models to the given datasets. For hyperparameter tuning, GridSearchCV class is used to evaluate various combinations of parameters.

3. Explore ensemble approach

There are 247 rows with the target variable ('area') being 0 out of 517, which is sufficient to say that the dataset is imbalanced. Chan [2] reports that ensemble models can perform well on imbalance datasets, therefore, exploring other approaches has a potential to find better generalised and rigorous models. For this task, BaggingRegressor, AdaBoostRegressor, and StackingRegressor (KneighborsRegressor and DecisionTreeRegressor classes are base estimators, and LinearRegression is the final estimator) classes will be tested.

ii) How the proposed solution is different from existing ones.

As described in i) 1, 2, and 3, following points are the difference from the ones in the literature:

- Different datasets
- Tuned hyperparameter
- Different classes

iii) Detail description of the model including all parameters so that any reader can implement your model.

Tested models with parameters are as follows:

Proposed models	Parameters
Decision tree	Criterion: absolute_error, Max_depth: 2
Random forest	Criterion: absolute_error, Max_depth: 3, max_features: sqrt, n_estinators: 500
Bagging regressor	Max_samples: 1.0, n_estimators: 50
AdaBoost regressor	Learning_rate: 1.0, n_estimators: 50
Stacking regressor	Knn__n_neighbors: 6, tree__max_depth: 5

iv) Description of experimental protocol.

For the data imbalanced data in the regression model, this task explores synthetic minority over-sampling technique for regression with gaussian noise (SMOGN) influenced by [3]. SMOGN is the combination of synthetic minority over-sampling technique for regression (SMOTER) with traditional interpolation, also with the introduction of gaussian noise (SMOTER-GN), using two over-sampling techniques determined by the KNN distances. This technique helps regression problems with imbalance datasets where the values in interest of predicting models are not enough volume. SMOGN technique is also used as an alternative experiment to log transformation where a skewed target value.
Datasets in [1] ('STFWI', 'STM', 'FWI', and 'M'), and a proposed dataset from feature importance analysis is transformed by smogn.smoter class developed in [4].

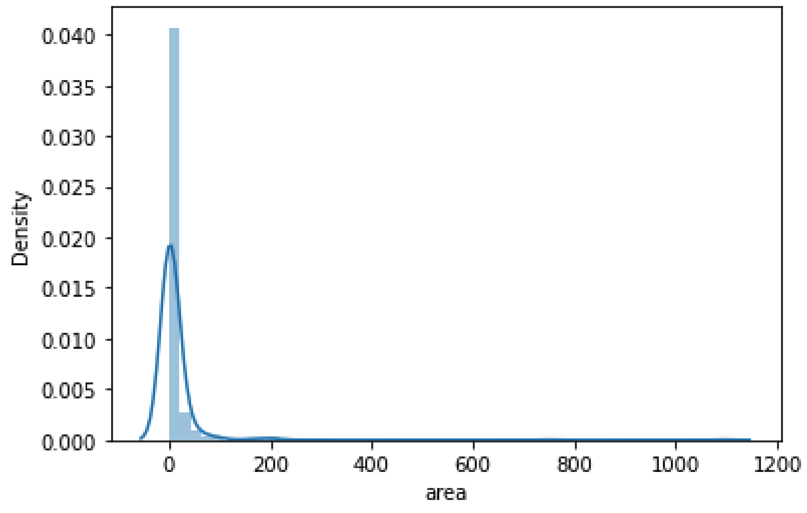
Data transformation by SMOGN (smogn.smoter)

```
data_smogn = smogn.smoter(
    data = data,
    y = 'area'
)
data_smogn.reset_index(drop=True, inplace=True)
```

The transformed data (data_smogn DataFrame) is less skewed compared to the original dataset (data DataFrame) shown as below graphs, and the dataset has a potential to provide better metrics:

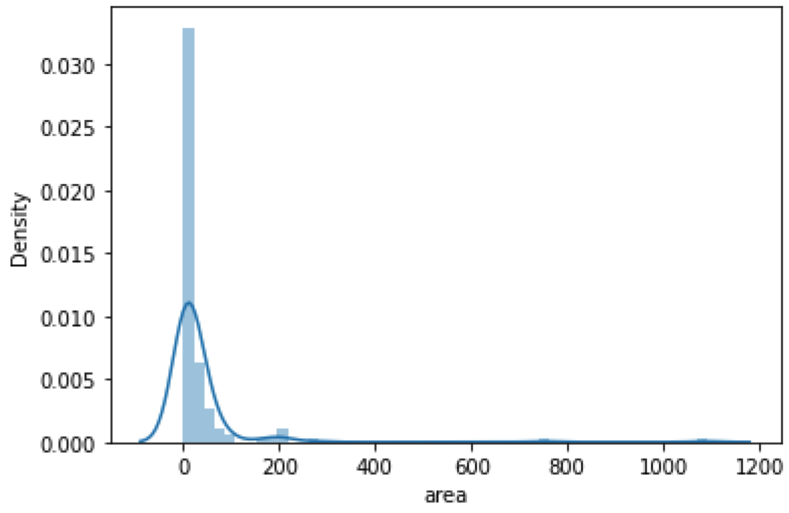
```
sns.distplot(data.area)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efbe6d83d50>
```



```
sns.distplot(data_smogn.area)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efbe6fac0d0>
```



As a preliminary experiment, transformed datasets were tested with models in [1] and results show underperformed, therefore, experimental protocol with SMOGN will not be used for the final solution.

	STFWI	STM	FWI	M	Feature Importance
Naive	81.676	83.183	82.453	84.102	85.450
Multiple regression	82.578	83.272	85.206	86.495	85.826
Decision tree	85.999	85.107	83.857	85.240	84.150
Random forest	84.820	85.309	83.458	81.951	83.136

v) Evaluation metrics.

The metrics to evaluate proposed solutions are RMSE, root mean squared error.

vi) Present results using tables and graphs.

The table below shows the evaluation metrics, RMSE, in each dataset and proposed mode.

	STFWI	STM	FWI	M	Feature importance
Decision tree	48.181	46.427	47.430	47.551	45.453
Random forest	46.793	46.107	46.125	47.637	47.216
Bagging regressor	46.669	46.910	45.418	**43.612**	45.702
AdaBoost regressor	46.858	46.516	**44.774**	46.940	47.059
Stacking regressor	**46.496**	**46.051**	46.052	44.708	**45.267**

vii) Compare and discuss results with respect to existing literature.

The best metrics in each datasets are as follows:

- STFWI: Support Vector machines (45.424)
- STM: Stacking regressor (46.051)
- FWI: AdaBoost regressor (44.774)
- M: Bagging regressor (43.612)
- Feature importance: Stacking regressor (45.267)

Ensemble models outperformed in three datasets (STM, FWI, and M) out of four original ones, which is reasonable to infer that ensemble models have more potential to explore for this task.
For FWI and M datasets, although metrics are varied (between around 44.0 and 47.5) in each model, AdaBoost regressor and Bagging regressor show improved results with 44.774 and 43.612 respectively, therefore, features in both datasets (‘FFMC’, ‘DMC’, ‘DC’, ‘ISI’) and (‘temp’, ‘RH’, ‘wind’, ‘rain’) could have better insight into this problem.

As an experiment, data transformation by SMOGN and a new dataset with proposed features based on feature importance analysis were tested, but they do not show improved performance compared to the original datasets.

viii) Appropriate references (IEEE numbered).

- [1] P. Cortez and A. Morais, "A Data Mining Approach to Predict Forest Fires using Meteorological Data", 2007. [Accessed 4 June 2022].
- [2] Y. Chan, *An Introduction to Approaches and Modern Applications with Ensemble Learning*. New York: Nova Science Publishers, Incorporated, 2021.
- [3] P. Branco, L. Torgo, and R. P. Ribeiro, “Pre-processing approaches for imbalanced distributions in regression,” *Neurocomputing*, vol. 343, pp. 76–99, May 2019, doi: 10.1016/j.neucom.2018.11.100.
- [4] N. Kunz, “nickkunz/smogn,” *GitHub*, Jun. 02, 2022. https://github.com/nickkunz/smogn (accessed Jun. 04, 2022).