Learning Systems Project Report

Venkata Sai Vivek Uddagiri

Introduction

The project is based upon classification and regression algorithms of machine learning which are used for classification and prediction of data. The classification and regression algorithms are both categorized under the same umbrella of supervised machine learning. Both share the similar concept of utilizing the trained datasets to predict the output, but there is a major difference between them i.e. the predicted output of a classification algorithm is categorical (discrete) and whereas the predicted output of a regression algorithm is numerical (continuous).

The classification algorithm attempts to estimate the mapping function (f) from the input variables (x) to discrete or categorize the output (y). The (y) is the category predicted by the mapping function (f). For example, when provided with a dataset about houses, a classification algorithm can try to predict whether the prices for the houses "sell more or less than the recommended retail price". The houses will be classified in whether the prices fall into two discrete categories i.e. above or below the said price. Commonly used classification algorithms include Logistic Regression, Naïve Bayes, Decision Tress and K Nearest Neighbors.

The regression algorithm attempts to estimate the mapping function (f) from the input variables (x) to numerical or continuous output (y). The (y) is a real value, which can be an integer or a floating point value. For example, when provided with a dataset about houses, a regression algorithm will try to predict the price of the house. Commonly used regression algorithms include linear regression, Support Vector Regression (SVR), and regression trees.

In this project, we are using six different datasets, where three datasets consist of classification problem and the other three datasets consist of regression problem.

METHOD:

1. For all the datasets we are converting the data into data frames. To do so we are using pandas. We are converting it in order to use built in libraries (seaborn.pairplot()) 2. We are visualizing the data by using libraries such as seaborn and matplotlib.

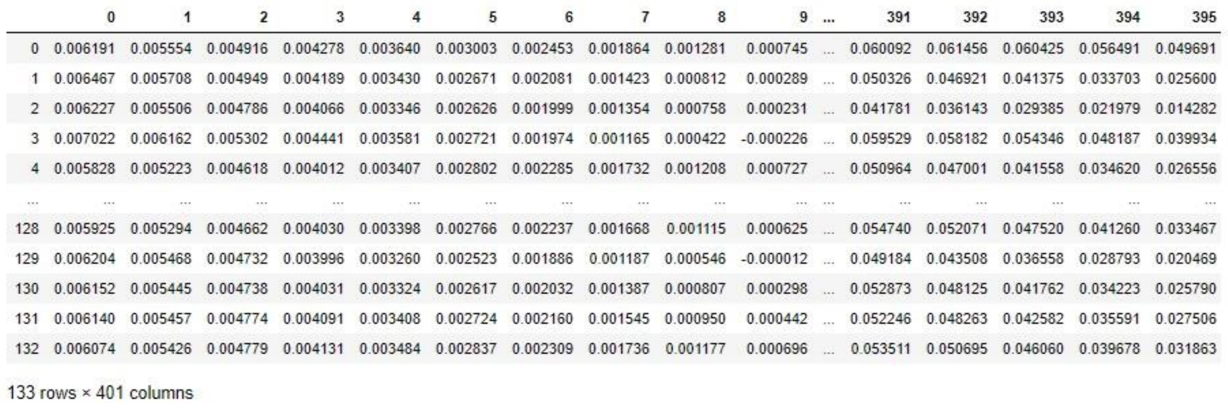
3. After visualization, we are normalizing the data using Standard Scalar and min max Scalar
   Standard Scalar – Every feature will be converted in the range of 0 to 1.
   Min Max Scalar – It converts the data into the range of -∞ to ∞.
4. We are visualizing the data before and after normalization; using seaborn.kdeplot().
5. We are checking generalization error by using cross-validation, for regression we are taking scoring as negative mean squared error and for classification we are taking scoring as accuracy. For regression, we are making a list of regression models and checking cross-validation score by iterating it and taking mean and standard deviation of cross-validation score, then checking which model gives us low error rate. For classification, we are performing similar tasks as that of regression, by using classification algorithms.
6. **Parameter tuning**, for SVM we are using Grid Search CV to tune our parameters like C(regularization parameter), Kernel ('rbf'', 'linear', 'poly') and gamma for polynomial kernel.

7. For KNN, we are iterating for different neighbors, calculating accuracy for classification, error for regression and visualizing using matplotlib inline. Through visualization we are checking which neighbor is giving us best result.
8. For two classification datasets (thyroid and electrocardiograms), we are using Deep Neural Networks. In thyroid there are three different classes of data (113, 260, 4627), when we are taking the complete data for machine learning algorithms, it was predicting only third class as output because it has more amount of data. To overcome this problem, we used two different techniques, one is down sampling; taking less data of third class, second technique is using neural networks and trying different threshold values for the network and we are getting accuracy by tuning threshold values. The threshold values are the median value of the data. The electrocardiogram consist of two different classes 0 and 1, we are using SVM and kernel as 'rbf'.
9. In neural networks, we are using optimizer as Adam and activation functions relu and sigmoid, loss as categorical cross-entropy (thyroid) and binary cross-entropy (electrocardiogram), and metric as accuracy.
10. For calculating the final result, we are taking root mean squared error (regression) which gives us the exact error for our algorithm and r^2 score which tells us how much our algorithm is best fit. For classification, we are calculating accuracy score, which gives us accuracy of our test result and classification report tells us the precision, recall and F1 score.

DATA:

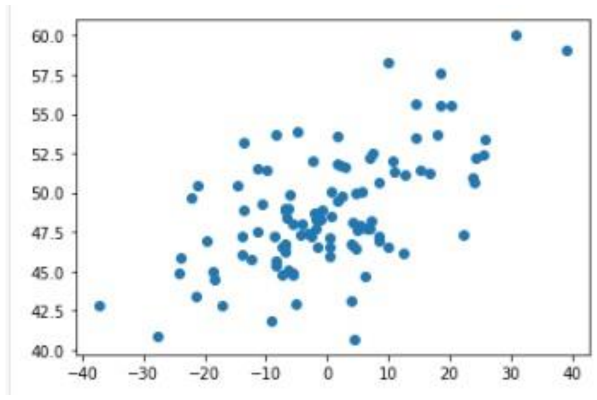
For all the datasets, we are extracting their best features by using PCA technique 1.

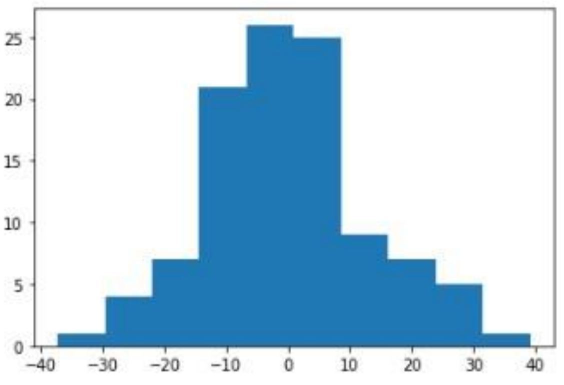
Estimating Cetane number of diesel fuel (Regression):



	0	1	2	3	4	5	6	7	8	9	...	391	392	393	394	395
0	0.006191	0.005554	0.004916	0.004278	0.003640	0.003003	0.002453	0.001864	0.001281	0.000745	...	0.060092	0.061456	0.060425	0.056491	0.049691
1	0.006467	0.005708	0.004949	0.004189	0.003430	0.002671	0.002081	0.001423	0.000812	0.000289	...	0.050326	0.046921	0.041375	0.033703	0.025600
2	0.006227	0.005506	0.004786	0.004066	0.003346	0.002626	0.001999	0.001354	0.000758	0.000231	...	0.041781	0.036143	0.029385	0.021979	0.014282
3	0.007022	0.006162	0.005302	0.004441	0.003581	0.002721	0.001974	0.001165	0.000422	-0.000226	...	0.059529	0.058182	0.054346	0.048187	0.039934
4	0.005828	0.005223	0.004618	0.004012	0.003407	0.002802	0.002285	0.001732	0.001208	0.000727	...	0.050964	0.047001	0.041558	0.034620	0.026556
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
128	0.005925	0.005294	0.004662	0.004030	0.003398	0.002766	0.002237	0.001668	0.001115	0.000625	...	0.054740	0.052071	0.047520	0.041260	0.033467
129	0.006204	0.005468	0.004732	0.003996	0.003260	0.002523	0.001886	0.001187	0.000546	-0.000012	...	0.049184	0.043508	0.036558	0.028793	0.020469
130	0.006152	0.005445	0.004738	0.004031	0.003324	0.002617	0.002032	0.001387	0.000807	0.000298	...	0.052873	0.048125	0.041762	0.034223	0.025790
131	0.006140	0.005457	0.004774	0.004091	0.003408	0.002724	0.002160	0.001545	0.000950	0.000442	...	0.052246	0.048263	0.042582	0.035591	0.027506
132	0.006074	0.005426	0.004779	0.004131	0.003484	0.002837	0.002309	0.001736	0.001177	0.000696	...	0.053511	0.050695	0.046060	0.039678	0.031863

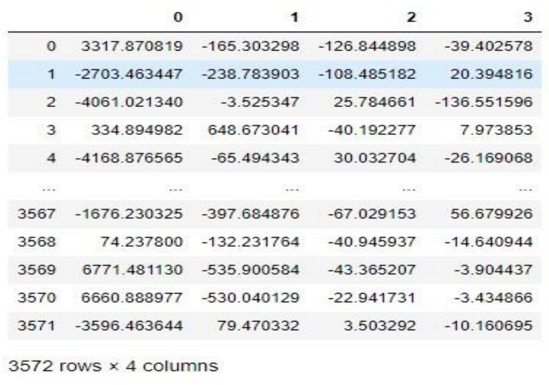
133 rows × 401 columns





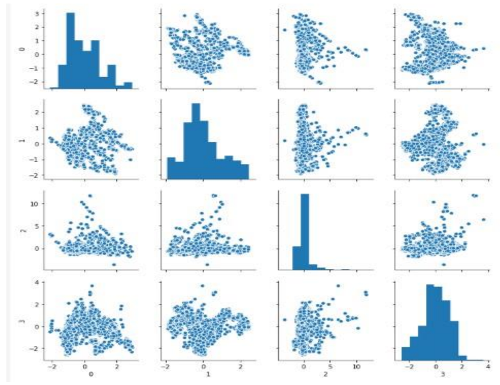
In this dataset we are taking only one component, from the above figures we can say that the data is completely scattered and Gaussian distributed.

2. Modeling the need for cooling in a H2O2 process (Regression)



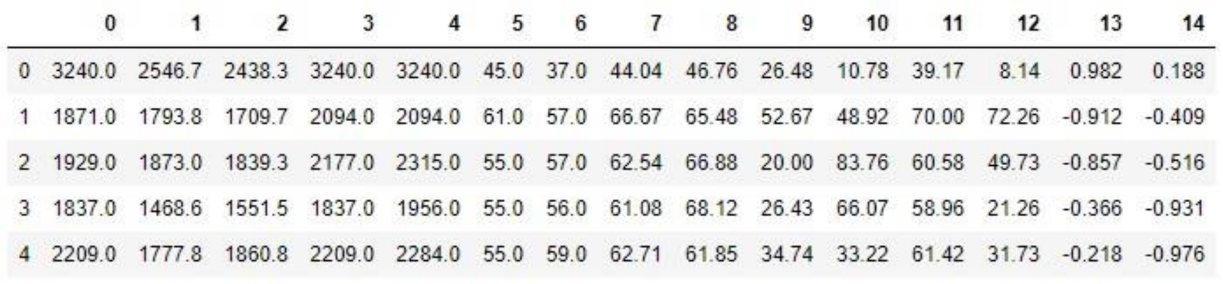
	0	1	2	3
0	3317.870819	-165.303298	-126.844898	-39.402578
1	-2703.463447	-238.783903	-108.485182	20.394816
2	-4061.021340	-3.525347	25.784661	-136.551596
3	334.894982	648.673041	-40.192277	7.973853
4	-4168.876565	-65.494343	30.032704	-26.169068
...	...	...	...	...
3567	-1676.230325	-397.684876	-67.029153	56.679926
3568	74.237800	-132.231764	-40.945937	-14.640944
3569	6771.481130	-535.900584	-43.365207	-3.904437
3570	6660.888977	-530.040129	-22.941731	-3.434866
3571	-3596.463644	79.470332	3.503292	-10.160695

3572 rows × 4 columns

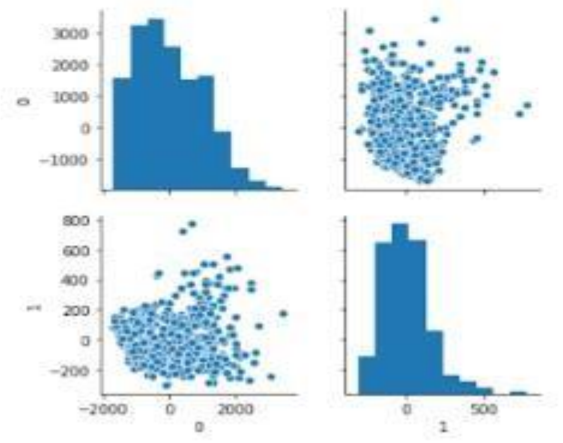


Extracting best features and plotting every feature using pairplot().

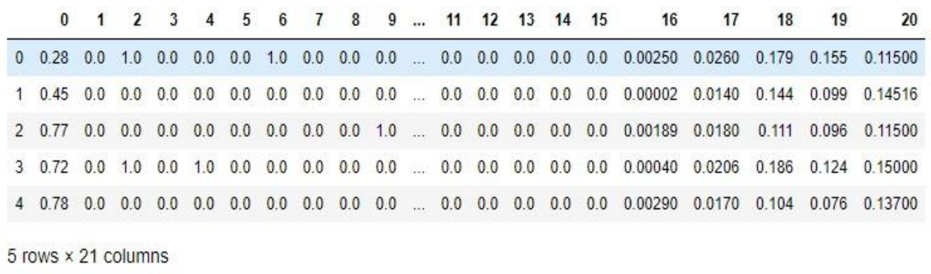
3. Predicting power load (Regression):



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3240.0	2546.7	2438.3	3240.0	3240.0	45.0	37.0	44.04	46.76	26.48	10.78	39.17	8.14	0.982	0.188
1	1871.0	1793.8	1709.7	2094.0	2094.0	61.0	57.0	66.67	65.48	52.67	48.92	70.00	72.26	-0.912	-0.409
2	1929.0	1873.0	1839.3	2177.0	2315.0	55.0	57.0	62.54	66.88	20.00	83.76	60.58	49.73	-0.857	-0.516
3	1837.0	1468.6	1551.5	1837.0	1956.0	55.0	56.0	61.08	68.12	26.43	66.07	58.96	21.26	-0.366	-0.931
4	2209.0	1777.8	1860.8	2209.0	2284.0	55.0	59.0	62.71	61.85	34.74	33.22	61.42	31.73	-0.218	-0.976

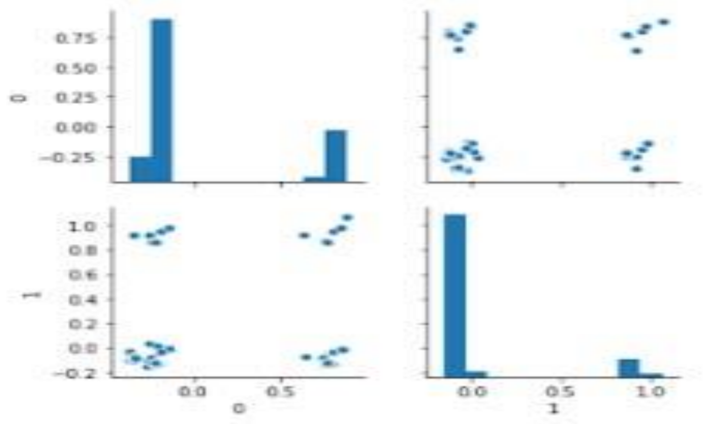


4. Thyroid disease (Classification):

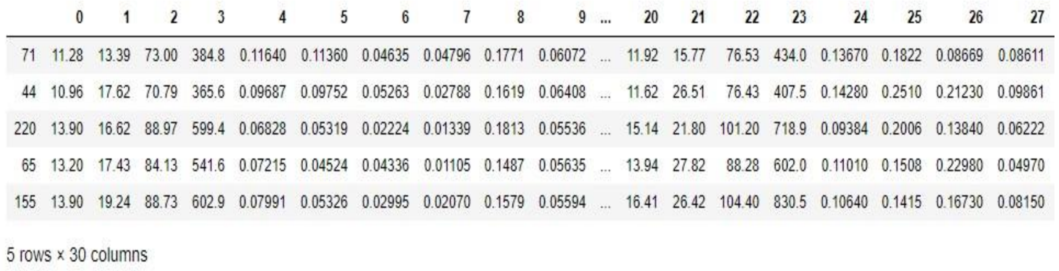


	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	15	16	17	18	19	20
0	0.28	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00250	0.0260	0.179	0.155	0.11500
1	0.45	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00002	0.0140	0.144	0.099	0.14516
2	0.77	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.00189	0.0180	0.111	0.096	0.11500
3	0.72	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00040	0.0206	0.186	0.124	0.15000
4	0.78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.00290	0.0170	0.104	0.076	0.13700

5 rows × 21 columns

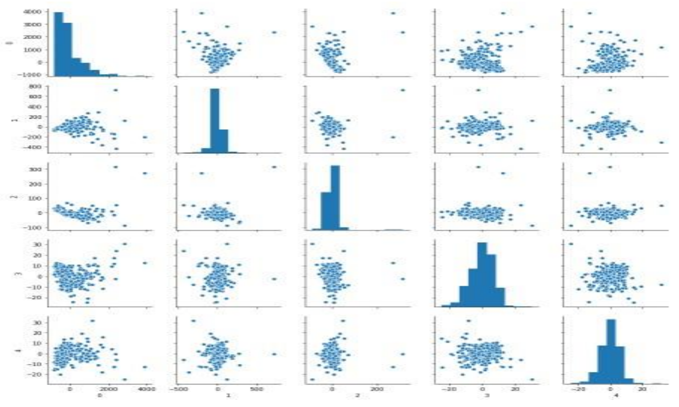


5. Breast cancer (Classification):

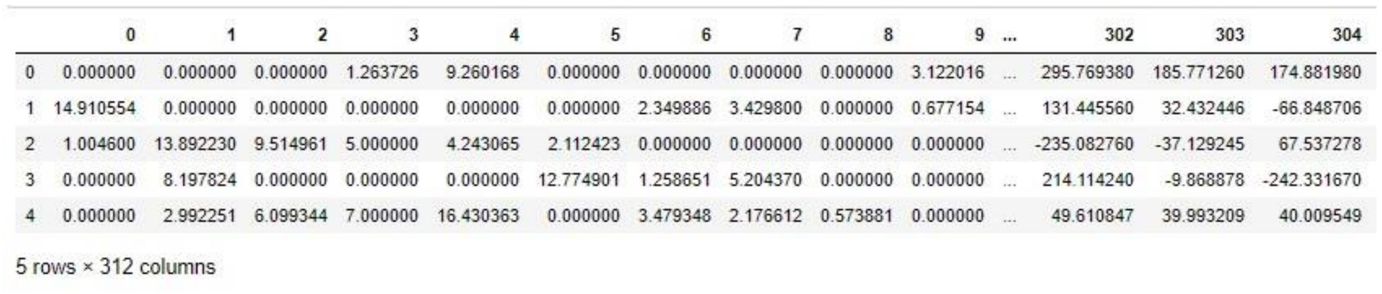


	0	1	2	3	4	5	6	7	8	9	...	20	21	22	23	24	25	26	27
71	11.28	13.39	73.00	384.8	0.11640	0.11360	0.04635	0.04796	0.1771	0.06072	...	11.92	15.77	76.53	434.0	0.13670	0.1822	0.08669	0.08611
44	10.96	17.62	70.79	365.6	0.09687	0.09752	0.05263	0.02788	0.1619	0.06408	...	11.62	26.51	76.43	407.5	0.14280	0.2510	0.21230	0.09861
220	13.90	16.62	88.97	599.4	0.06828	0.05319	0.02224	0.01339	0.1813	0.05536	...	15.14	21.80	101.20	718.9	0.09384	0.2006	0.13840	0.06222
65	13.20	17.43	84.13	541.6	0.07215	0.04524	0.04336	0.01105	0.1487	0.05635	...	13.94	27.82	88.28	602.0	0.11010	0.1508	0.22980	0.04970
155	13.90	19.24	88.73	602.9	0.07991	0.05326	0.02995	0.02070	0.1579	0.05594	...	16.41	26.42	104.40	830.5	0.10640	0.1415	0.16730	0.08150

5 rows × 30 columns

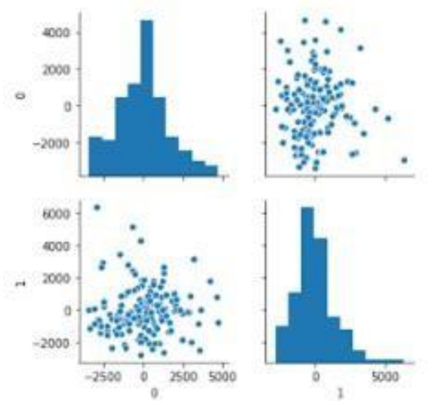


6. Electrocardiograms (Classification):



	0	1	2	3	4	5	6	7	8	9	...	302	303	304
0	0.000000	0.000000	0.000000	1.263726	9.260168	0.000000	0.000000	0.000000	0.000000	3.122016	...	295.769380	185.771260	174.881980
1	14.910554	0.000000	0.000000	0.000000	0.000000	0.000000	2.349886	3.429800	0.000000	0.677154	...	131.445560	32.432446	-66.848706
2	1.004600	13.892230	9.514961	5.000000	4.243065	2.112423	0.000000	0.000000	0.000000	0.000000	...	-235.082760	-37.129245	67.537278
3	0.000000	8.197824	0.000000	0.000000	0.000000	12.774901	1.258651	5.204370	0.000000	0.000000	...	214.114240	-9.868878	-242.331670
4	0.000000	2.992251	6.099344	7.000000	16.430363	0.000000	3.479348	2.176612	0.573881	0.000000	...	49.610847	39.993209	40.009549

5 rows × 312 columns



RESULTS:

Most of machine learning algorithms are distance based algorithms, they calculate the distance between the predicted line and our actual data point, so we need to normalize our data to get optimal result.

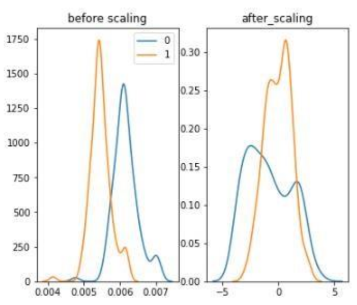


Fig1

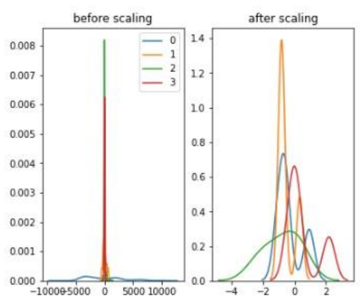


Fig2

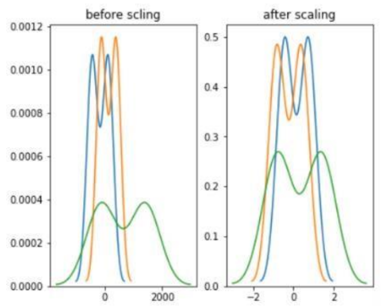


Fig3

Where Fig1: Estimating Cetane number of diesel fuel (Regression)

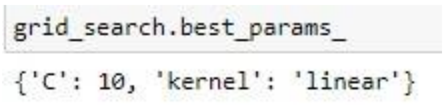
Fig2: Modeling the need for cooling in a H2O2 process (Regression)

Fig3: Predicting power load (Regression)

From the above figures we can say that before scaling there is a big difference between in each feature value, we are scaling the data in order to get the data in a single scale range for all the features, so that we can obtain better results.

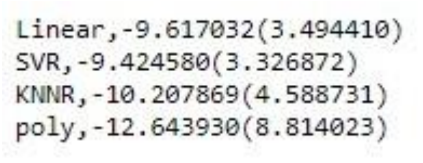
1. Estimating Cetane number of diesel fuel (Regression):

We are hyper tuning Grid Search CV to obtain the best parameter for SVM, we got below parameters as best parameters and performing SVM using these parameters.



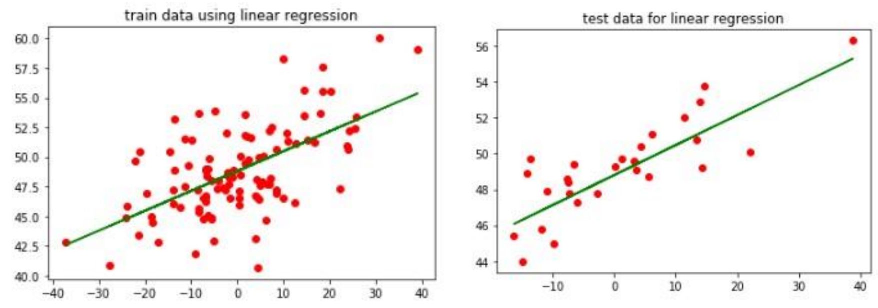
```
grid_search.best_params_
{'C': 10, 'kernel': 'linear'}
```

We are calculating generalization error rate and finding the mean and standard deviation of the error and we can observe from the below error rates, that SVR and Linear Regression are getting lowest error rate with low standard deviation value, so we are training our data using SVR algorithm.

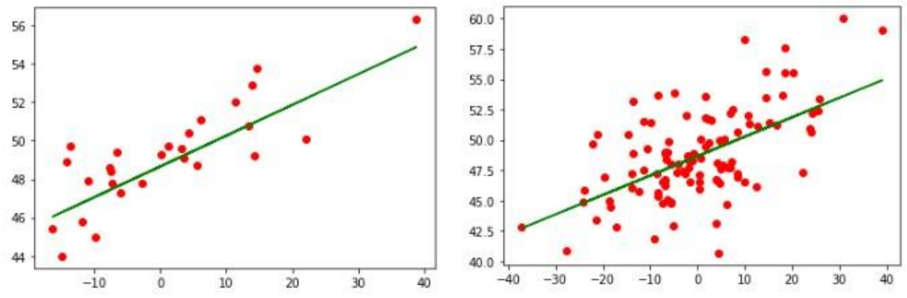


```
Linear,-9.617032(3.494410)
SVR,-9.424580(3.326872)
KNNR,-10.207869(4.588731)
poly,-12.643930(8.814023)
```

The below figures are train data and test data for Linear Regression, we are calculating root mean squared error and r^2 score. The green line is the best fit line.



The root mean squared error and r^2 score for test data are 1.4966316804051227 and 0.670362613 8623306.
The root mean squared error and r^2 score for train data are 2.9993843746627733 and 0.3530377495138569
The below figures are train and test data for SVM, using linear kernel.



The left figure is test data and right figure is train data.
test data root mean square error: 1.534596437705426, r2_score of SVR using linear kernel: 0.653 4268089460884
We can say that by using SVM we are getting less root mean squared error with r^2 as 65%, which indicates that SVM is the best algorithm.

2. Modeling the need for cooling in a H2O2 process (Regression)

We are hyper tuning Grid Search CV to obtain the best parameter for SVM, we got below parameters as best parameters and performing SVM using these parameters.

```
grid_search.best_params_
```

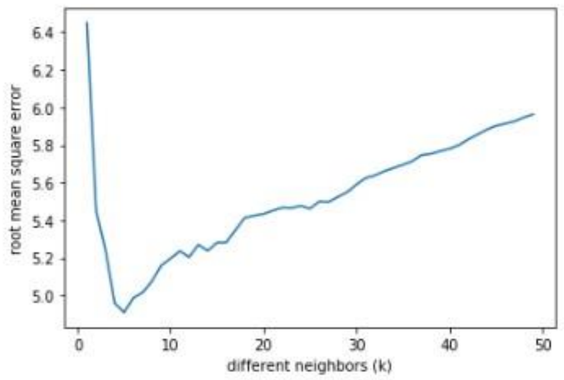
```
{'C': 1000, 'gamma': 0.8, 'kernel': 'rbf'}
```

We are calculating generalization error rate and finding the negative mean squared error.

```
linear regression:-84.878600(10.628244)
KNeighborsRegressor:-28.468811(6.852757)
SVR:-35.929105(9.402654)
RandomForestRegressor:-85.569082(10.775971)
DecisionTreeRegressor:-50.082890(12.404267)
Polynomial_regression:-84.878600(10.628244)
```

We can observe that KNN is giving less error rate compared to other algorithms.

```
Text(0, 0.5, 'root mean square error')
```



We are calculating errors for different neighbors, we can observe that for n = 5 we are getting low error i.e. 4.909998684372218 and r^2 : 0.8113419742134151.

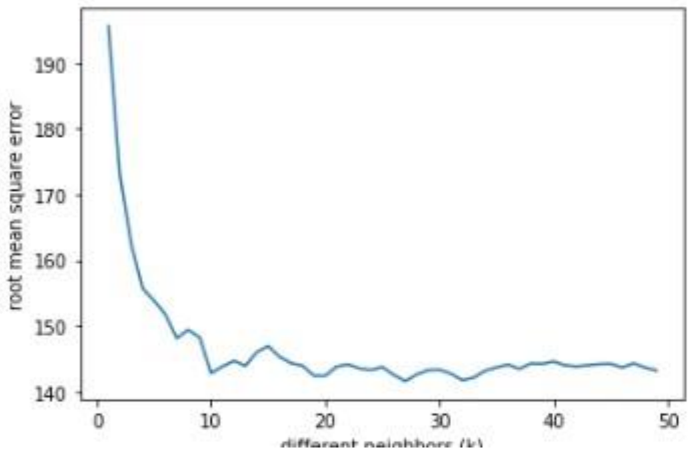
3. Predicting power load (Regression)

This dataset is a non-linear dataset, hence we are using non-linear models for calculating negative mean squared error.

```
SVR:-203453.480314(33377.747120)
KNeighborsRegressor:-24636.205901(6262.850307)
Polynomial_regression:-19031.958009(4299.729039)
```

We can observe that Polynomial regression model is giving low error compared to other models.

```
Text(0, 0.5, 'root mean square error')
```



While we are calculating for different neighbors we get the above figure. We can observe that for n = 10 we are getting low error: 148.21985307227192 r^2 : 0.9174176742173048

When we are training our data using polynomial regression with degree = 2, we are getting Error = 141.58977179588072 and r^2 = 0.9246404812137429

We can say that polynomial regression is giving us best result.

For classification, most of machine learning algorithms are distance based algorithms, they calculate the distance between the predicted line and our actual data point, so we need to normalize our data to get optimal result.

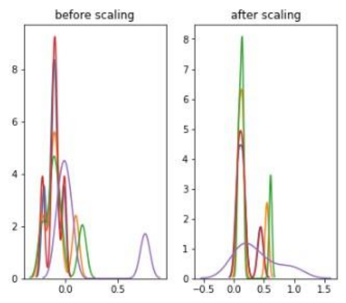


Fig4

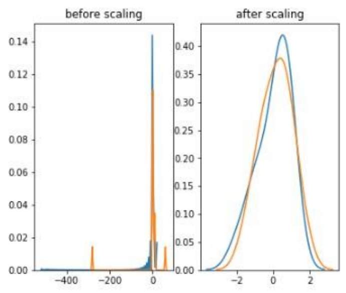


Fig5

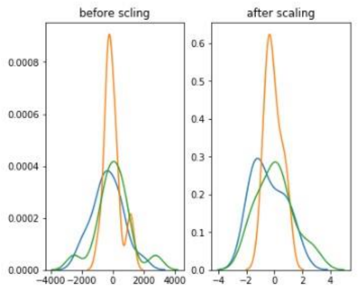


Fig6

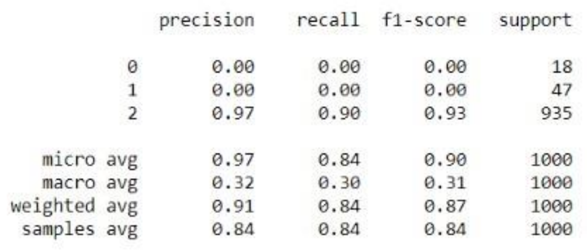
Where Fig4: Thyroid disease (Classification)

Fig5: Breast Cancer (Classification)

Fig6: Electrocardiogram (Classificaiton)

4. Thyroid Disease:

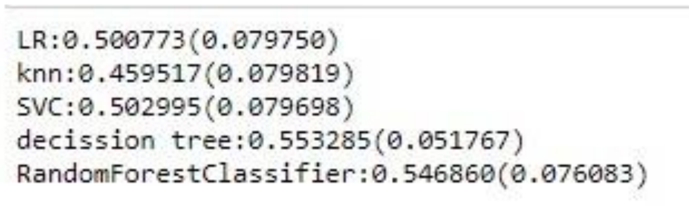
In this dataset we have three different classes, when we use whole date for ML algorithm it is only predicting only one class



```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        18
           1       0.00      0.00      0.00        47
           2       0.97      0.90      0.93       935

   micro avg       0.97      0.84      0.90      1000
   macro avg       0.32      0.30      0.31      1000
weighted avg       0.91      0.84      0.87      1000
 samples avg       0.84      0.84      0.84      1000
```

To overcome this, we are using two different techniques down sampling and using different threshold values.

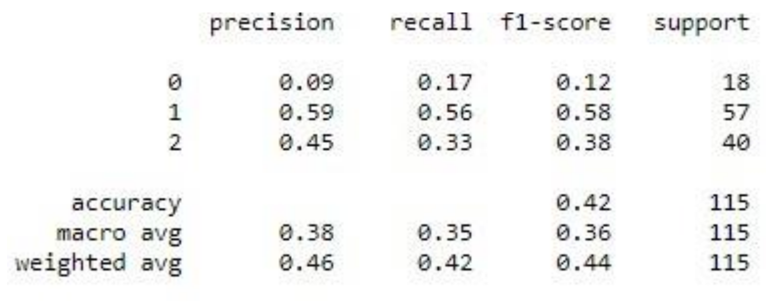
For down sampling



```
LR:0.500773(0.079750)
knn:0.459517(0.079819)
SVC:0.502995(0.079698)
decission tree:0.553285(0.051767)
RandomForestClassifier:0.546860(0.076083)
```

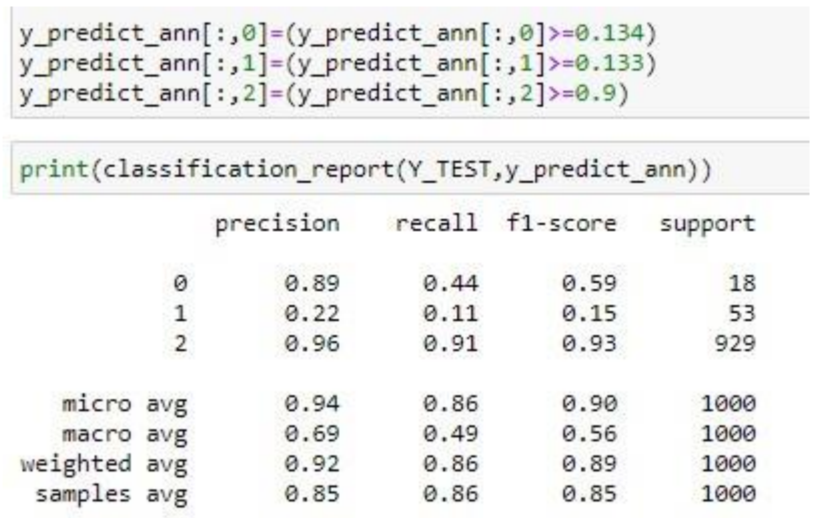
We can observe that decision tree is giving us the highest accuracy.

For Decision tree our accuracy = 41% and precision, recall and f1-score are as follows:



```
              precision    recall  f1-score   support

           0       0.09      0.17      0.12        18
           1       0.59      0.56      0.58        57
           2       0.45      0.33      0.38        40

    accuracy                           0.42       115
   macro avg       0.38      0.35      0.36       115
weighted avg       0.46      0.42      0.44       115
```

The second method: we are changing threshold values and setting threshold value as median value, by doing this we are getting better accuracy then ML algorithms.



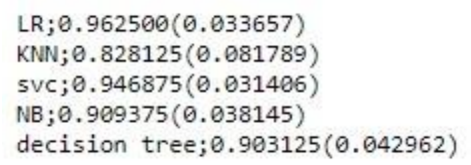
```
y_predict_ann[:,0]=(y_predict_ann[:,0]>=0.134)
y_predict_ann[:,1]=(y_predict_ann[:,1]>=0.133)
y_predict_ann[:,2]=(y_predict_ann[:,2]>=0.9)
```

```
print(classification_report(Y_TEST,y_predict_ann))
```

```
              precision    recall  f1-score   support

           0       0.89      0.44      0.59        18
           1       0.22      0.11      0.15        53
           2       0.96      0.91      0.93       929

   micro avg       0.94      0.86      0.90      1000
   macro avg       0.69      0.49      0.56      1000
weighted avg       0.92      0.86      0.89      1000
 samples avg       0.85      0.86      0.85      1000
```

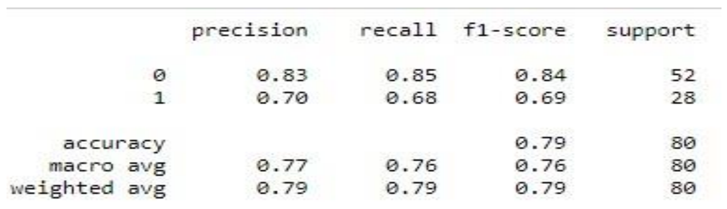
5. Breast Cancer:

In this dataset we have two different classes 0 and 1, we are calculating generalization error using cross validation.



```
LR;0.962500(0.033657)
KNN;0.828125(0.081789)
svc;0.946875(0.031406)
NB;0.909375(0.038145)
decision tree;0.903125(0.042962)
```

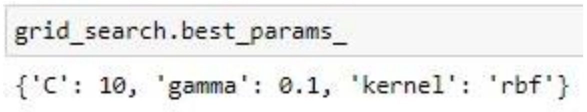
We can observe that logistic regression is giving better accuracy, the classification report is as follows:



```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84        52
           1       0.70      0.68      0.69        28

    accuracy                           0.79        80
   macro avg       0.77      0.76      0.76        80
weighted avg       0.79      0.79      0.79        80
```

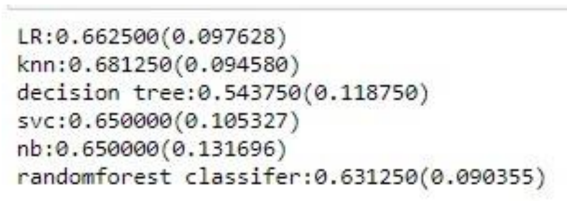
6. Electrocardiogram:

In this dataset, we are getting the best parameters for SVM as follows:



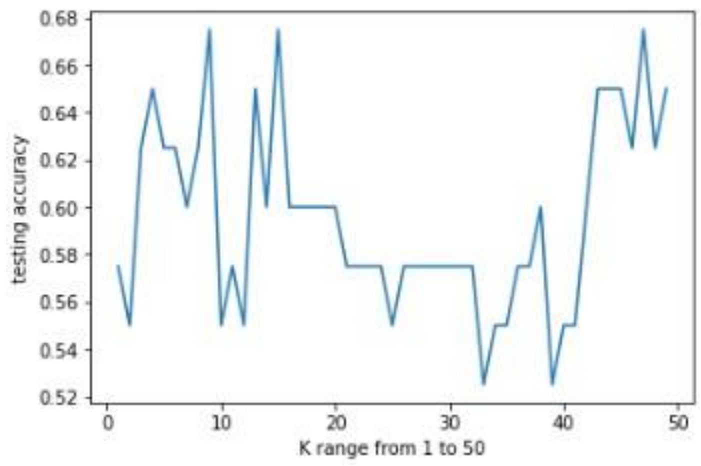
```
grid_search.best_params_

{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

We are calculating cross-validation score as follows:



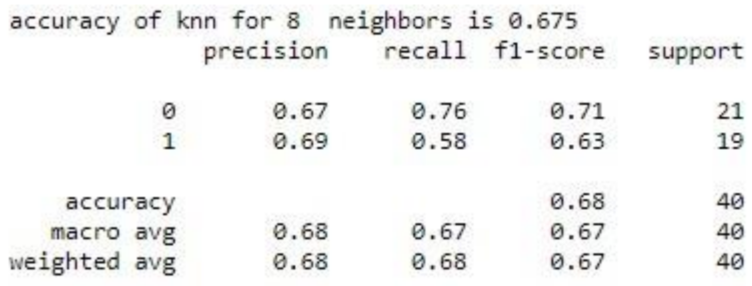
```
LR:0.662500(0.097628)
knn:0.681250(0.094580)
decision tree:0.543750(0.118750)
svc:0.650000(0.105327)
nb:0.650000(0.131696)
randomforest classifer:0.631250(0.090355)
```

We can observe that KNN is giving us the highest accuracy.



The above figure is of KNN with different neighbors, we can observe that for n = 10 it is giving highest accuracy.

The classification report is as follows:



```
accuracy of knn for 8  neighbors is 0.675
              precision    recall  f1-score   support

           0       0.67      0.76      0.71        21
           1       0.69      0.58      0.63        19

    accuracy                           0.68        40
   macro avg       0.68      0.67      0.67        40
weighted avg       0.68      0.68      0.67        40
```

<u>**Conlusion:**</u>

Selecting best features and hyper parameter tuning in our dataset is very important in order to achieve optimal output.

The cross-validation plays a vital role in classification and regression, it tells us which algorithm is best suited for training and testing the data, and predicting its output optimally.
In classification if one class has very large data compared to other classes, the algorithm is predicting only that class, to overcome this we need to take measure and use methods such as down sampling or changing the threshold values to predict the other classes.

References:

ML models for classification and Regression:
o https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
• Feature Selection: o https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection
• Principal Component Analysis (PCA) for dimensionality reduction: o https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html o Examples: https://scikit-learn.org/stable/modules/decomposition.html#pca
• Model selection: https://scikit-learn.org/stable/model_selection.html o Cross-validation: https://scikit-learn.org/stable/modules/cross_validation.html
o https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
o Hyper-parameters tuning: https://scikit-learn.org/stable/modules/grid_search.html o Model evaluation: https://scikit-learn.org/stable/modules/model_evaluation.html o Validation curves: https://scikit-learn.org/stable/modules/learning_curve.html