

Configuration Manual

MSc Research Project
Data Analytics

Pavan Kumar Sudhakar
Student ID: X17126738

School of Computing
National College of Ireland

Supervisor: Dr.Christian Horn

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Pavan Kumar Sudhakar
Student ID:	X17126738
Programme:	Data Analytics
Year:	2018-2019
Module:	MSc Research Project
Supervisor:	Dr.Christian Horn
Submission Due Date:	12/08/2019
Project Title:	Configuration Manual
Word Count:	396
Page Count:	15

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input checked="" type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input checked="" type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input checked="" type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Pavan Kumar Sudhakar
X17126738

f

1 Software Requirement

The entire coding is done using python version 3. Anaconda needs to be installed on the system in order to use Python. Spyder IDE is used for programming in python. Anaconda can be downloaded from ¹. A 64 bit version of anaconda needs to be installed.

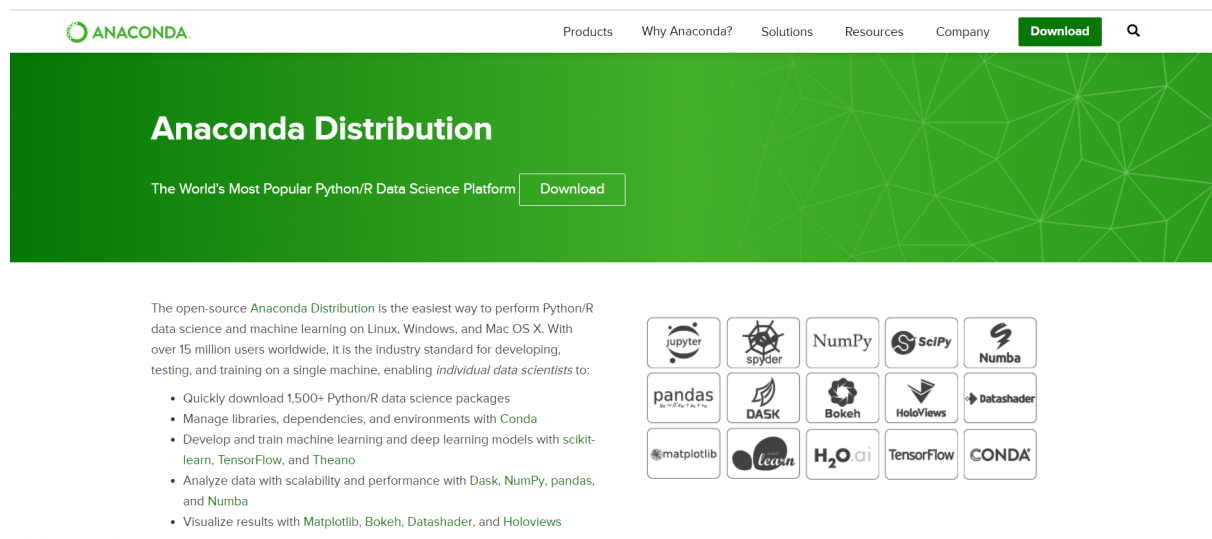


Figure 1: Downloading Anaconda

Anaconda Navigator should be opened from: **Start-Anaconda Navigator**

¹<https://www.anaconda.com/distribution/>

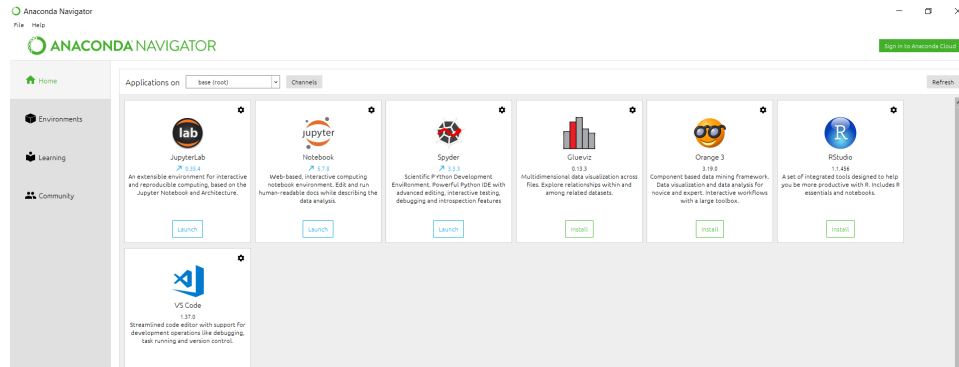


Figure 2: Downloading Anaconda

2 Data Extraction

The Energy dataset was downloaded from ². The dataset can be downloaded directly from the above link. Data needs to be very carefully integrated as there are missing values and missing rows.

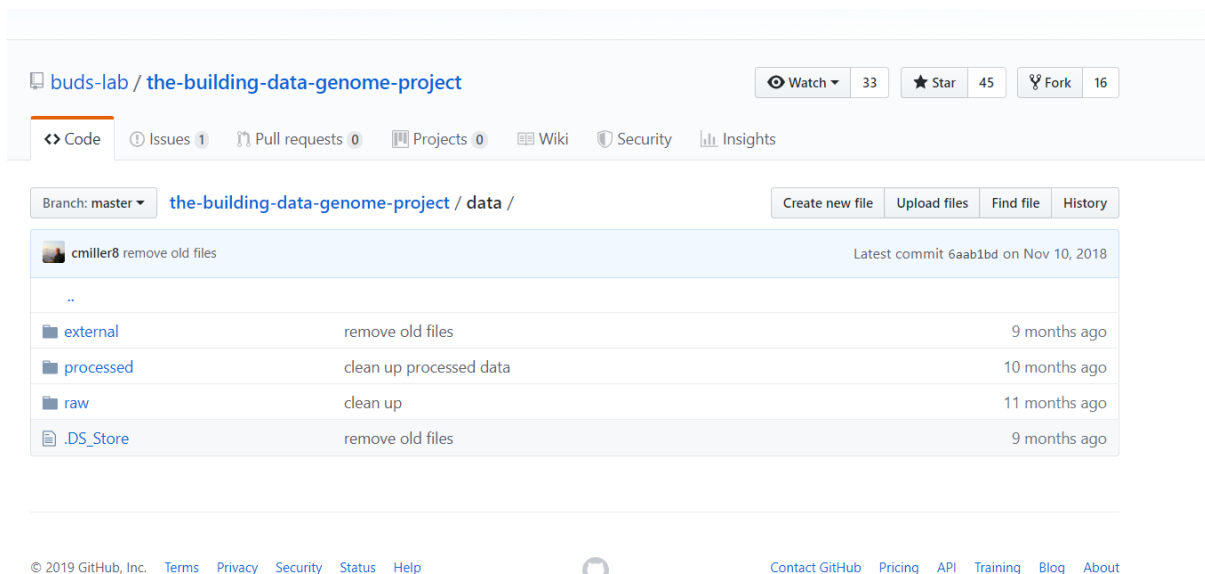


Figure 3: Downloading Raw Energy data

²<https://github.com/buds-lab/the-building-data-genome-project/tree/master/data>

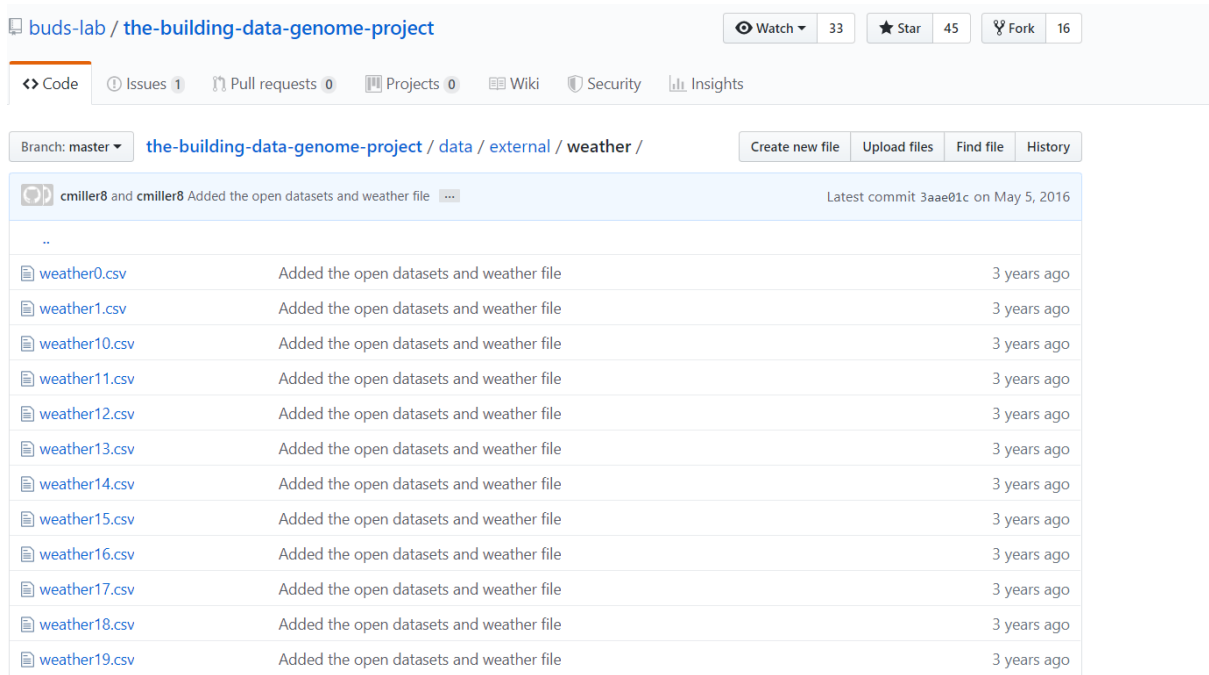


Figure 4: Downloading Weather data

3 Data Pre-Processing

Launch Spyder from **Start - Spyder** and open a new python3 console using spyder.

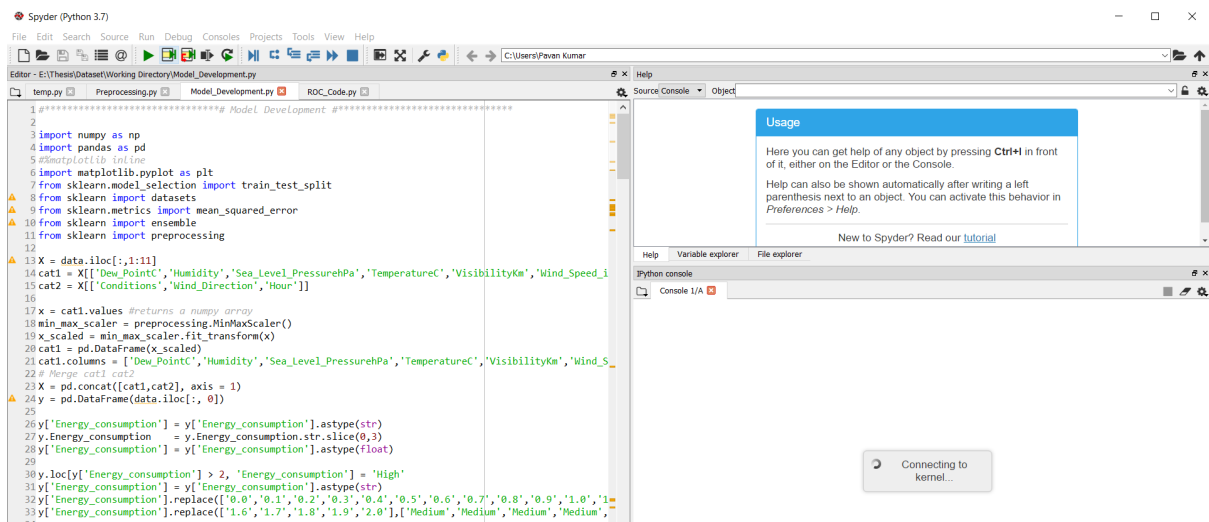


Figure 5: Spyder Console

Load the file pre-processing file to perform data import, cleaning, EDA and transformation.

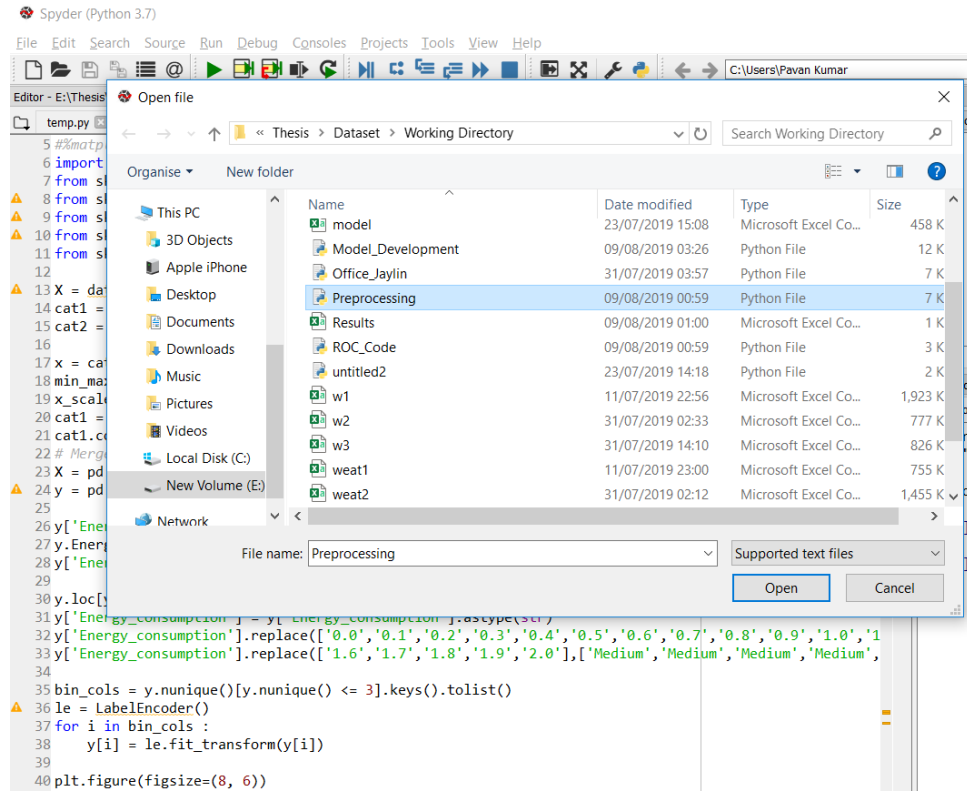


Figure 6: Spyder Console

Select and execute the codes at at once. Similarly load data for all buildings and execute one by one.

4 Library Installation

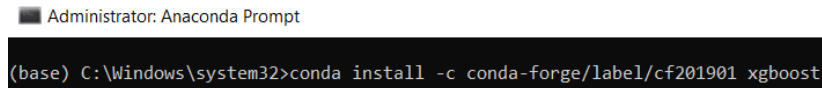
The following libraries needs to be installed in anaconda prompt.

1. XG Boost
2. Vecstack

4.1 Installing XG Boost

XG boost is installed by using any of the following command. Open Anaconda prompt in administer mode and execute any of the following commands.

- `conda install -c conda-forge xgboost`
- `conda install -c conda-forge/label/gcc7 xgboost`
- `conda install -c conda-forge/label/cf201901 xgboost`



```
Administrator: Anaconda Prompt
(base) C:\Windows\system32>conda install -c conda-forge/label/cf201901 xgboost
```

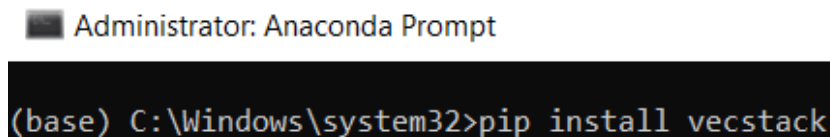
Figure 7: XG Boost Installation

Note: Some of the above command might not work on some machines, as it version specific.

4.2 Installing Vecstack

Similar to XG Boost, install vecstack using anaconda prompt using the below command.

- pip install vecstack



```
Administrator: Anaconda Prompt
(base) C:\Windows\system32>pip install vecstack
```

Figure 8: Vecstack Installation

5 Code listing

5.1 Pre-Processing

5.1.1 Co-Relation Plot

```
1
2 # Co-Relation Plot 1
3
4 def heatmap(x, y, size):
5     fig, ax = plt.subplots()
6
7     # Mapping from column names to integer coordinates
8     x_labels = [v for v in sorted(x.unique())]
9     y_labels = [v for v in sorted(y.unique())]
10    x_to_num = {p[1]:p[0] for p in enumerate(x_labels)}
11    y_to_num = {p[1]:p[0] for p in enumerate(y_labels)}
12
13    size_scale = 500
14    ax.scatter(
15        x=x.map(x_to_num),
16        y=y.map(y_to_num),
17        s=size * size_scale,
18        marker='s'
```

```

19 )
20 # Show column labels on the axes
21 ax.set_xticks([x_to_num[v] for v in x_labels])
22 ax.set_xticklabels(x_labels, rotation=45, horizontalalignment='
right')
23 ax.set_yticks([y_to_num[v] for v in y_labels])
24 ax.set_yticklabels(y_labels)
25
26 columns = ['Energy_consumption', 'Dew_PointC', 'TemperatureC', 'Humidity
', 'Sea_Level_PressurehPa', 'VisibilityKm', 'Wind_Speed_in_KMperHour'
, 'WindDirDegrees', 'Hour']
27 corr = data[columns].corr()
28 corr = pd.melt(corr.reset_index(), id_vars='index') # Unpivot the
dataframe, so we can get pair of arrays for x and y
29 corr.columns = ['x', 'y', 'value']
30 heatmap(
31     x=corr['x'],
32     y=corr['y'],
33     size=corr['value'].abs()
34 )

```

SVC

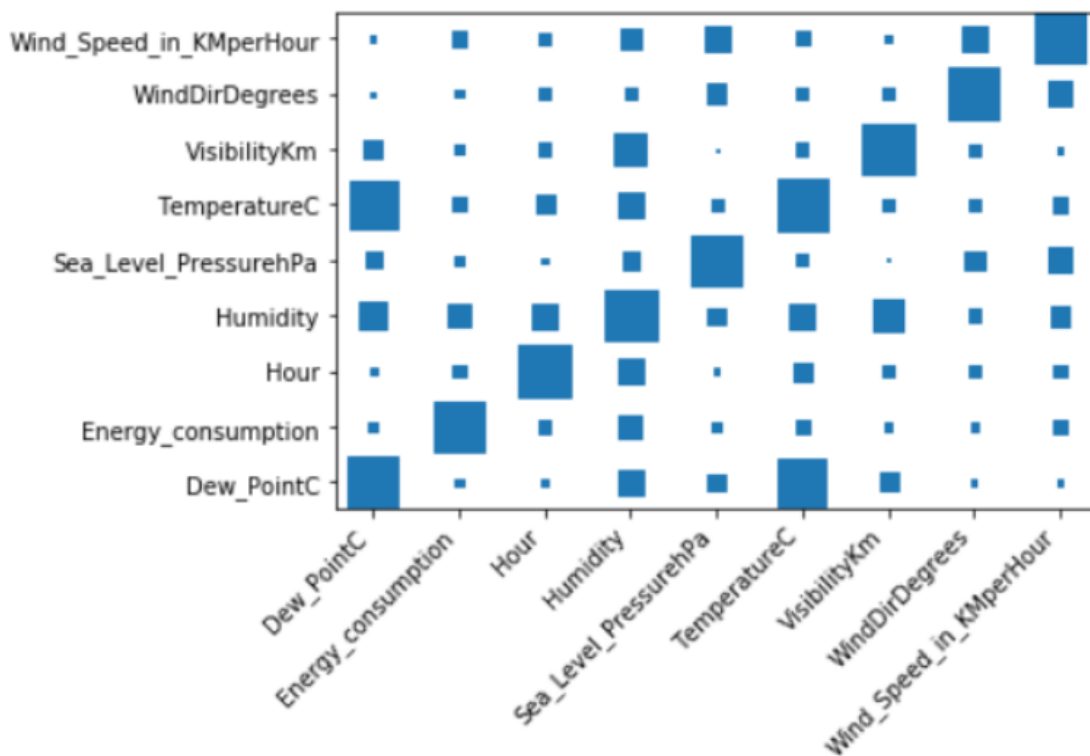


Figure 9: XG Boost Installation

5.1.2 Class Imbalance Plots

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import numpy as np

```



```

4 # Plotting the classes of dependent variable
5 plt.figure(figsize=(8, 6))
6 sns.countplot('Conditions', data=data)
7 plt.title('Balanced Classes')
8 plt.show()
9
10 # Plotting the classes of dependent variable
11 plt.figure(figsize=(8, 6))
12 sns.countplot('Wind_Direction', data=data)
13 plt.title('Balanced Classes')
14 plt.show()

```

SVC

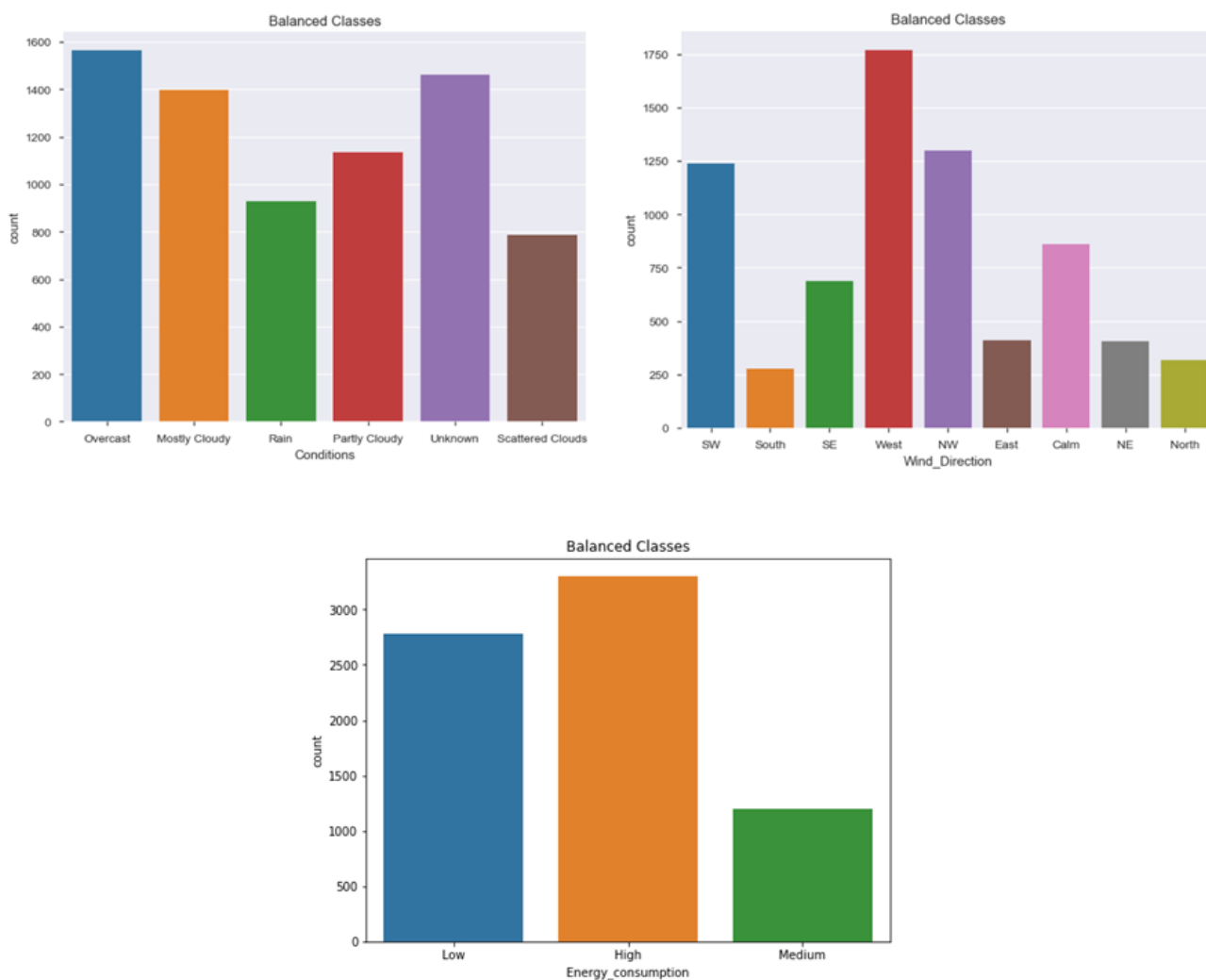


Figure 10: XG Boost Installation

5.1.3 Variable Distribution

```

1 plot = plotdata.iloc[:,1]
2 sns.distplot(plot);
3 plot = plotdata.iloc[:,2]

```

```

4 sns.distplot(plot, hist=False, rug=True);
5 plot = plotdata.iloc[:,3]
6 sns.kdeplot(plot, shade=True);
7 plot = plotdata.iloc[:,4]
8 sns.distplot(plot, kde=False, fit=stats.gamma);
9 plot = plotdata.iloc[:,5]
10 sns.distplot(plot, kde=False, fit=stats.gamma);
11 plot = plotdata.iloc[:,6]
12 sns.kdeplot(plot, shade=True);
13 plot = plotdata.iloc[:,7]
14 sns.kdeplot(plot, shade=True);
15 plot = plotdata.iloc[:,8]
16 sns.kdeplot(plot, shade=True);
17 plot = plotdata.iloc[:,9]
18 sns.kdeplot(plot, shade=True);
19 plot = plotdata.iloc[:,10]
20 sns.kdeplot(plot, shade=True);

```

SVC

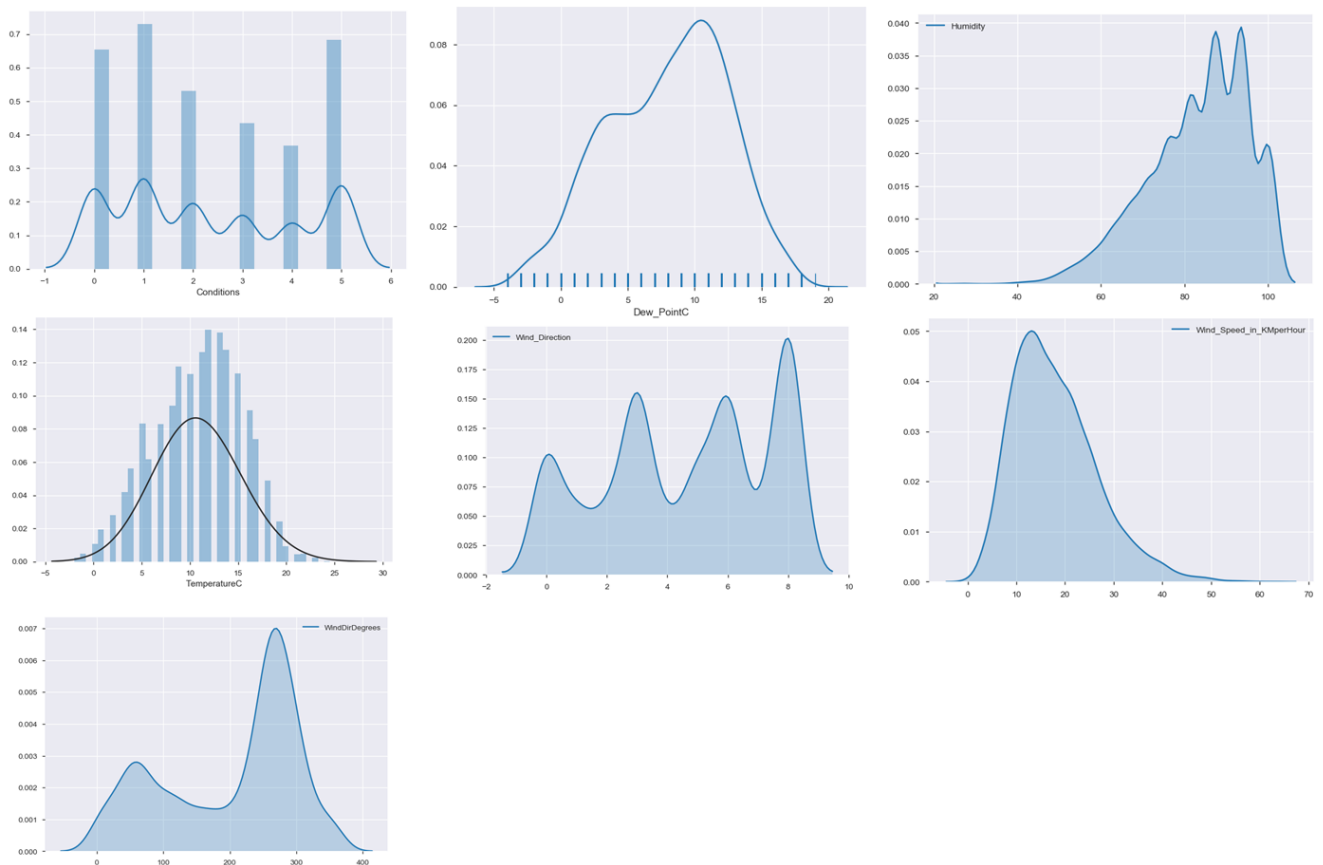


Figure 11: XG Boost Installation

5.1.4 Label Encoding

```
1 from sklearn.preprocessing import LabelEncoder
2 bin_cols = data.nunique()[data.nunique() <= 9].keys().tolist()
3 le = LabelEncoder()
4 for i in bin_cols :
5     data[i] = le.fit_transform(data[i])
6
7
8 bin_cols = data.nunique()[data.nunique() <= 9].keys().tolist()
9 le = LabelEncoder()
10 for i in bin_cols :
11     data[i] = le.fit_transform(data[i])
12 data.Wind_Direction.unique()
```

SVC

5.2 Algorithm Generation

Load the modeldeployment python file and run all at once. The below code snippets shows the code for algorithm generation.

5.2.1 SVC Algorithm generation

```
1 # SVC
2 from sklearn.metrics import classification_report
3 from sklearn.svm import SVC
4 svc = SVC(C = 100, kernel = 'rbf', gamma = 0.001, random_state = 0)
5 svc.fit(x_training_set, y_training_set.values.ravel())
6 y_pred = svc.predict(x_test_set)
7 #accuracy_score(y_pred,y_test_set)
8 print("Accuracy score (training): {0:.3f}".format(svc.score(
9     x_training_set, y_training_set)))
9 print("Accuracy score (validation): {0:.3f}".format(svc.score(
10     x_test_set, y_test_set)))
10 print()
11 evaluation(y_test_set, y_pred)
```

SVC

5.2.2 XG Boost Algorithm generation

```
1 xgb = XGBClassifier(learning_rate =0.43,
2 booster = 'gbtree', # 'dart 69%'
3 n_estimators=100,
4 max_depth=26,
5 min_child_weight=1,
6 gamma=0,
7 subsample=0.8,
8 colsample_bytree=0.8,
9 objective= 'binary:logistic',
10 nthread=4,
11 scale_pos_weight=1,
12 verbosity =1,
13 seed=27)
14 xgb.fit(x_training_set, y_training_set.values.ravel())
```

```

15
16 # Predicting the Test set results
17 y_pred = xgb.predict(x_test_set)
18 #accuracy_score(y_pred,y_test_set)
19 accuracy = cross_val_score(estimator = xgb, X = x_training_set, y =
    y_training_set.values.ravel(), cv =10)
20 accuracy.mean()
21 evaluation(y_test_set,y_pred)
22
23 from xgboost import plot_importance
24 plot_importance(xgb)

```

XG Boost

5.2.3 Gradient Boost

```

1 from sklearn.ensemble import GradientBoostingClassifier
2 from sklearn.metrics import classification_report, confusion_matrix,
    roc_curve, auc
3 gb = GradientBoostingClassifier(n_estimators=26, learning_rate = 0.28 ,
    max_features=2, max_depth = 19, random_state = 0)
4 gb.fit(x_training_set, y_training_set.values.ravel())
5 y_pred = gb.predict(x_test_set)
6
7 print("Accuracy score (training): {0:.3f}".format(gb.score(
    x_training_set, y_training_set)))
8
9 print("Accuracy score (validation): {0:.3f}".format(gb.score(x_test_set
    , y_test_set)))
10 print()

```

Gradient Boost

5.2.4 Ada Boost

```

1 from sklearn.ensemble import AdaBoostClassifier
2 from sklearn.tree import DecisionTreeClassifier
3 adb = AdaBoostClassifier(
4     DecisionTreeClassifier(max_depth=51),
5     n_estimators= 200
6 )
7 adb.fit(x_training_set, y_training_set.values.ravel())
8 y_pred = adb.predict(x_test_set)

```

Ada Boost

5.2.5 Proposed Hybrid Ensembled model

```

1 from vecstack import stacking
2 from sklearn.svm import SVC
3 import pandas as pd
4 from sklearn.metrics import accuracy_score
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.ensemble import RandomForestClassifier
7 from sklearn.ensemble import AdaBoostClassifier
8 from sklearn.tree import DecisionTreeClassifier
9 from xgboost import XGBClassifier

```

```

10 from vecstack import stacking
11 models = [
12     SVC(C = 100, kernel = 'rbf', gamma = 0.001, random_state = 0),
13     GradientBoostingClassifier(n_estimators=26, learning_rate = 0.28 ,
14     max_features=2, max_depth = 19, random_state = 0),
15     AdaBoostClassifier(DecisionTreeClassifier(max_depth=51),
16     n_estimators= 200)
17 ]
18 S_train, S_test = stacking(models,
19                             x_training_set, y_training_set, x_test_set,
20                             regression=False,
21                             mode='oof_pred_bag',
22                             needs_proba=False,
23                             save_dir=None,
24                             metric=accuracy_score,
25                             n_folds=10,
26                             stratified=True,
27                             shuffle=True,
28                             random_state=0,
29                             verbose=2)
30
31 model = XGBClassifier(learning_rate = 0.43, booster = 'gbtree',
32                       n_estimators=100, max_depth=26, min_child_weight=1,
33                       gamma=0, subsample=0.8, colsample_bytree=0.8, objective=
34                       'binary:logistic', nthread=4,
35                       scale_pos_weight=1, seed=27)
36 model = model.fit(S_train, y_training_set.values.ravel())
37
38 y_pred = model.predict(S_test)
39 print('Final prediction score: [%.8f]' % accuracy_score(y_test_set,
40 y_pred))

```

Ada Boost

5.2.6 Hyper Parameter - Code Listing

```

1 #Hyper paramater tuning starts for XGBClassifier
2
3 depth = np.arange(0,1,1)
4 estimators = np.arange(0,10,1)
5 rate = np.arange(0.1,1,0.1)
6
7 params_grid = {
8     'max_depth' : depth,
9     'n_estimators' : estimators,
10    'learning_rate' : rate
11 }
12 params_fixed = {
13     'objective' : 'binary:logistic',
14     'silent' :1
15 }
16 bst_grid = GridSearchCV(
17     estimator = XGBClassifier(**params_fixed, seed = seed),
18     param_grid = params_grid,
19     scoring = 'accuracy'
20 )
21 bst_grid.fit(x_training_set, y_training_set)
22 bst_grid.cv_results_

```

```

23 print("Best accuracy Obtained: {0}".format(bst_grid.best_score_))
24 print("Parameters")
25 for key, value in bst_grid.best_params_.items():
26     print("\t{0}:{1}".format(key, value))

```

Stacked Ensemble Model

6 Results

6.1 AUC

The results of the algorithm can be verified using accuracy score obtained. Also, by using AUC curve for better understanding. An code snippet of AUC curve generation is shown below.

```

1 from itertools import cycle
2 from sklearn.metrics import roc_curve, auc
3 from sklearn.preprocessing import label_binarize
4 from sklearn.multiclass import OneVsRestClassifier
5 from scipy import interp
6
7 y = label_binarize(y, classes=[0, 1, 2])
8 n_classes = y.shape[1]
9
10 # Learn to predict each class against the other
11
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =.10, random_state=50)
13
14 classifier = OneVsRestClassifier(GradientBoostingClassifier(
    n_estimators=26, learning_rate = 0.28 , max_features=2, max_depth =
    19, random_state = 0))
15
16 y_score = classifier.fit(X_train, y_train).decision_function(X_test)
17
18 fpr = dict()
19 tpr = dict()
20 roc_auc = dict()
21 for i in range(n_classes):
22     fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
23     roc_auc[i] = auc(fpr[i], tpr[i])
24
25 fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel
    ())
26 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
27
28 plt.figure()
29 lw = 2
30 plt.plot(fpr[2], tpr[2], color='darkorange',
31         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[2])
32 plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
33 plt.xlim([0.0, 1.0])
34 plt.ylim([0.0, 1.05])
35 plt.xlabel('False Positive Rate')
36 plt.ylabel('True Positive Rate')
37 plt.title('Receiver operating characteristic example')

```

```

38 plt.legend(loc="lower right")
39 plt.show()
40
41 all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
42
43 mean_tpr = np.zeros_like(all_fpr)
44 for i in range(n_classes):
45     mean_tpr += interp(all_fpr, fpr[i], tpr[i])
46
47 mean_tpr /= n_classes
48
49 fpr["macro"] = all_fpr
50 tpr["macro"] = mean_tpr
51 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
52
53 # Plot all ROC curves
54 plt.figure()
55 plt.plot(fpr["micro"], tpr["micro"],
56         label='micro-average AUC curve (area = {0:0.2f})'
57             ''.format(roc_auc["micro"]),
58         color='deeppink', linestyle=':', linewidth=4)
59
60 plt.plot(fpr["macro"], tpr["macro"],
61         label='macro-average AUC curve (area = {0:0.2f})'
62             ''.format(roc_auc["macro"]),
63         color='navy', linestyle=':', linewidth=4)
64
65 colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
66 for i, color in zip(range(n_classes), colors):
67     plt.plot(fpr[i], tpr[i], color=color, lw=lw,
68             label='AUC curve of class {0} (area = {1:0.2f})'
69                 ''.format(i, roc_auc[i]))
70
71 plt.plot([0, 1], [0, 1], 'k--', lw=lw)
72 plt.xlim([0.0, 1.0])
73 plt.ylim([0.0, 1.05])
74 plt.xlabel('False Positive Rate')
75 plt.ylabel('True Positive Rate')
76 plt.title('Receiver operating characteristic to multi-class using Micro
77           , macro averaging')
78 plt.legend(loc="lower right")
79 plt.show()

```

AUC generation

Receiver operating characteristic to multi-class using Micro, macro averaging

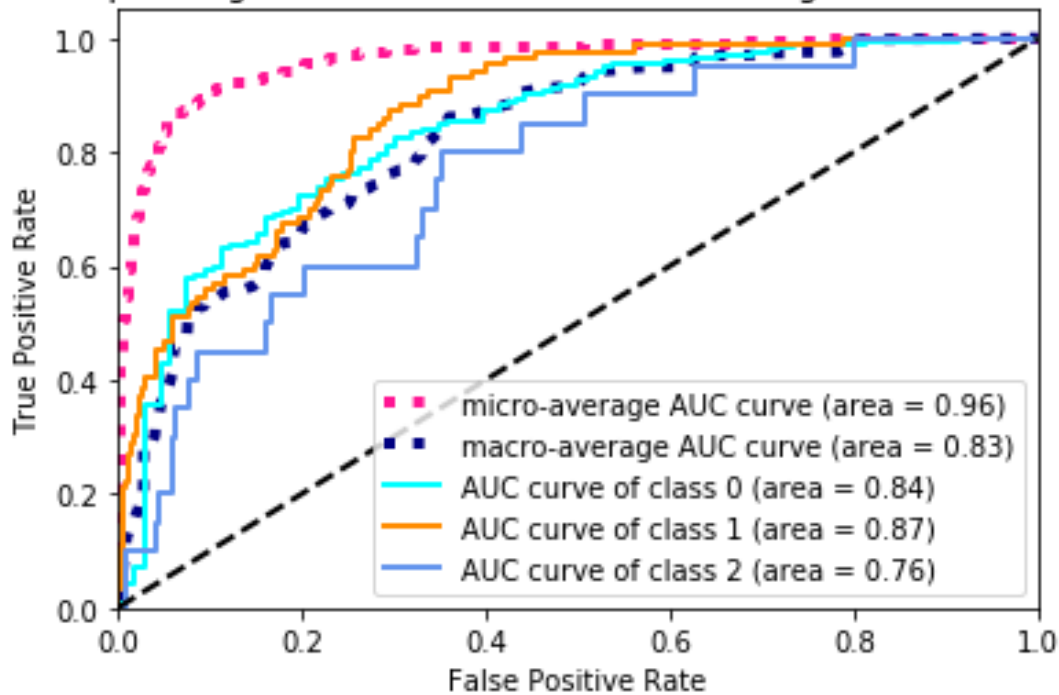


Figure 12: XG Boost Installation

6.2 Mann-Whitney test and table generation

```
1 from scipy import stats
2 print(stats.shapiro(output))
3 output.columns = ['Stack','xgb','gb','svc','adb','gnb']
4
5 # P-Value of 0.002 indicates that the data is not normally distributed,
6 #hence using Mann-Whitney U-test to check for statistical significance
7
8 stats.mannwhitneyu(output['Stack'],output['gnb'],use_continuity=True,
9 alternative=None)
10 stats.mannwhitneyu(output['Stack'],output['svc'],use_continuity=True,
11 alternative=None)
12 stats.mannwhitneyu(output['Stack'],output['adb'],use_continuity=True,
13 alternative=None)
14 stats.mannwhitneyu(output['Stack'],output['xgb'],use_continuity=True,
15 alternative=None)
16 stats.mannwhitneyu(output['Stack'],output['gb'],use_continuity=True,
17 alternative=None)
18
19 results =
20 [0.0000903,0.0000729,0.00693772344142142,0.09244142720655457,0.454742198871193]
21
22 results = pd.DataFrame(results)
23 results['Algorithm'] = ['SVC','naivebayes','Adaboost','XGBoost','
24 Gradient']
25 results['Significance'] = ['Yes','Yes','Yes','No','No']
```

AUC generation

Matplotlib: Python plotting Matplotlib 3.1.1 documentation (n.d.) *NumPy NumPy* (n.d.) *Python Data Analysis Library pandas: Python Data Analysis Library* (n.d.) *scikit-learn: machine learning in Python scikit-learn 0.21.3 documentation* (n.d.) *seaborn: statistical data visualization seaborn 0.9.0 documentation* (n.d.) *Ivanov (n.d.) XGBoost Documentation xgboost 1.0.0-SNAPSHOT documentation* (n.d.) *SciPy.org SciPy.org* (n.d.)

References

Ivanov, I. (n.d.). vecstack: Python package for stacking (machine learning technique).

URL: <https://github.com/vecxoz/vecstack>

Matplotlib: Python plotting Matplotlib 3.1.1 documentation (n.d.).

URL: <https://matplotlib.org/>

NumPy NumPy (n.d.).

URL: <https://www.numpy.org/>

Python Data Analysis Library pandas: Python Data Analysis Library (n.d.).

URL: <https://pandas.pydata.org/>

scikit-learn: machine learning in Python scikit-learn 0.21.3 documentation (n.d.).

URL: <https://scikit-learn.org/stable/>

SciPy.org SciPy.org (n.d.).

URL: <https://www.scipy.org/>

seaborn: statistical data visualization seaborn 0.9.0 documentation (n.d.).

URL: <https://seaborn.pydata.org/>

XGBoost Documentation xgboost 1.0.0-SNAPSHOT documentation (n.d.).

URL: <https://xgboost.readthedocs.io/en/latest/>