Configuration Manual of Research Project

MSc Research Project
Data Analytics

Prajwal Shashidhara
Student ID: x22209077

School of Computing
National College of Ireland

Supervisor: Dr. Bharat Agarwal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Prajwal Shashidhara
Student ID:	x22209077
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Bharat Agarwal
Submission Due Date:	16/09/2024
Project Title:	Configuration Manual of Research Project
Word Count:	371
Page Count:	12

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:	Prajwal Shashidhara
Date:	14th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
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.	☐

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 of Research Project: Predictive Modelling for Cost Estimation in Construction Projects Using Machine Learning Algorithms

Prajwal Shashidhara
x22209077

1 Introduction

The given configuration manual corresponds to the different configuration steps used that generate the results for the study "Predictive Modelling for Cost Estimation in Construction Projects Using Machine Learning Algorithms". The following states all the configuration techniques, software and libraries used to build the code and achieve state-of-the-art results

2 System specifications

- Device name DESKTOP-FRQ155L
- Processor Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
- Installed RAM 20.0 GB (19.9 GB usable)
- System type 64-bit operating system, x64-based processor
- Edition Windows 11 Pro Version 23H2

3 Software Requirements

- Anaconda 3
- Python 3.10
- Jupyter Notebook

4 Python Libraries

Major libraries used in the research are:

- Numpy
- Pandas
- Seaborn
- Matplotlib
- Plotly
- Sklearn
- StatsModel
- Keras
- SHAP

Libraries

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import shap
shap.initjs()
%matplotlib inline

import scipy.stats as stats
import statsmodels.formula.api as smf

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import MissingIndicator, SimpleImputer
from sklearn.preprocessing import  PolynomialFeatures, KBinsDiscretizer, FunctionTransformer
from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, LabelBinarizer, OrdinalEncoder
from sklearn.feature_selection import RFE,SelectKBest,f_classif
from sklearn.linear_model import LogisticRegression, LinearRegression, ElasticNet, Lasso, Ridge
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier, BaggingRegressor,RandomForestClassifier,RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier,GradientBoostingRegressor, AdaBoostClassifier, AdaBoostRegressor
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
from sklearn.svm import LinearSVC, LinearSVR, SVC, SVR
from sklearn.neural_network import MLPClassifier, MLPRegressor
from keras.models import Sequential
from keras.layers import Dense
import warnings
warnings.filterwarnings('ignore')
import imblearn
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, precision_score, recall_score, f1_score,classification_report
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import statsmodels.formula.api as smf
from sklearn.metrics import mean_absolute_error
```

Figure 1: Library import

5 Data Import

Data was imported using the panda's package, as shown in Figure 2

Data Ingestion

```
[3]: data = pd.read_excel('Residential-Building-Data-Set-new.xlsx')
```

Data Understanding

```
[6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 372 entries, 0 to 371
Columns: 109 entries, START YEAR to V-10
dtypes: float64(94), int64(15)
memory usage: 316.9 KB
```

```
[8]: data.head()
```

[8]:

	START YEAR	START QUARTER	COMPLETION YEAR	COMPLETION QUARTER	V-1	V-2	V-3	V-4	V-5	V-6	...	V-22.4	V-23.4	V-24.4	V-25.4	V-26.4	V-27.4	V-28.4	V-29.4	
0	81	1	85	1	1	3150.0	920.0	598.5	190	1010.84	...	815.50	1755.00	8002.0	60.74	54.26	2978.26	41407.0	601988.1	2
1	84	1	89	4	1	7600.0	1140.0	3040.0	400	963.81	...	1316.30	8364.78	8393.0	90.95	89.79	11379.37	44835.0	929027.1	5
2	78	1	81	4	1	4800.0	840.0	480.0	100	689.84	...	765.80	1755.00	4930.0	38.70	32.04	1653.06	37933.0	377828.6	1
3	72	2	73	2	1	685.0	202.0	13.7	20	459.54	...	152.25	1442.31	1456.0	9.73	8.34	686.16	8194.0	122031.7	
4	87	1	90	2	1	3000.0	800.0	1230.0	410	631.91	...	2244.10	9231.76	9286.0	136.60	140.20	9821.00	48260.0	1734973.5	5

5 rows × 109 columns

Figure 2: Data Import

6 Data filtering

All the important features are selected in the research based on the business problem,as shown in Figure 3

```
[12]: X = data.iloc[:,4:31]
      y = data.iloc[:,107:]
```

```
[14]: data = pd.concat([X,y],axis = 1)
```

```
[16]: data.head()
```

[16]:

	V-1	V-2	V-3	V-4	V-5	V-6	V-7	V-8	V-11	V-12	...	V-22	V-23	V-24	V-25	V-26	V-27	V-28	V-29	V-9	V-10
0	1	3150.0	920.0	598.5	190	1010.84	16	1200	6713.00	56.2	...	809.8	1755.00	8003.0	67.81	63.25	3758.77	42587.00	628132.9	2200	410
1	1	7600.0	1140.0	3040.0	400	963.81	23	2900	3152.00	106.0	...	1473.5	8842.18	8864.0	105.52	105.32	12113.01	45966.00	1188995.8	5000	1000
2	1	4800.0	840.0	480.0	100	689.84	15	630	1627.00	41.0	...	608.2	1755.00	7773.0	45.91	38.34	1537.96	39066.00	524764.8	1200	170
3	1	685.0	202.0	13.7	20	459.54	4	140	2580.93	12.1	...	211.1	1612.95	1649.0	11.62	10.06	392.96	8435.75	141542.6	165	30
4	1	3000.0	800.0	1230.0	410	631.91	13	5000	6790.00	203.8	...	3148.0	9248.40	9380.0	158.63	169.50	10082.00	49572.00	2318397.0	5500	700

5 rows × 29 columns

Figure 3: Filtering

7 Statistical Data Analysis and EDA

This section includes exploratory data analysis on the review columns. The distribution of topics across years is displayed in Figure 4.

```
[20]: def continuous_var_summary(x):
          return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),
                            x.std(), x.var(), x.min(), x.quantile(0.01), x.quantile(0.05),
                              x.quantile(0.10),x.quantile(0.25),x.quantile(0.50),x.quantile(0.75),
                                x.quantile(0.90),x.quantile(0.95), x.quantile(0.99),x.max()],
                          index = ['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1',
                                 'P5' ,'P10' ,'P25' ,'P50' ,'P75' ,'P90' ,'P95' ,'P99' ,'MAX'])
```

```
[22]: data.apply(continuous_var_summary).round(2).T
```

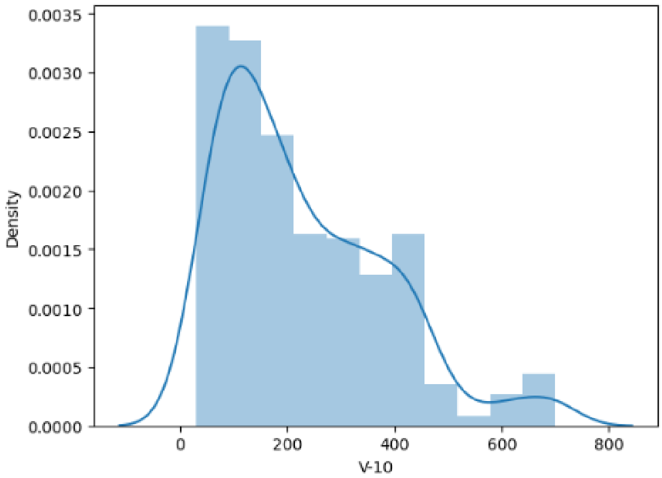
[22]:

	N	NMISS	SUM	MEAN	MEDIAN	STD	VAR	MIN	P1	P5	P10	P25	P50	P75	P9
V-1	372.0	0.0	3.619000e+03	9.73	8.00	6.56	4.308000e+01	1.00	1.00	1.00	2.00	4.00	8.00	17.00	19.
V-2	372.0	0.0	6.432045e+05	1729.04	1220.00	1802.37	3.248543e+06	200.00	258.40	370.00	451.00	720.00	1220.00	2100.00	3269.
V-3	372.0	0.0	1.585145e+05	426.11	300.00	490.08	2.401750e+05	60.00	80.00	100.00	120.00	190.00	300.00	490.50	786.
V-4	372.0	0.0	1.219803e+05	327.90	164.70	563.54	3.175822e+05	3.70	8.43	20.22	32.40	67.80	164.70	366.05	764.
V-5	372.0	0.0	6.068000e+04	163.12	140.00	112.60	1.267974e+04	10.00	10.00	30.00	40.00	80.00	140.00	230.00	309.
V-6	372.0	0.0	2.062442e+05	554.42	522.45	275.11	7.568329e+04	193.08	202.63	273.83	305.40	391.68	522.45	667.90	798.
V-7	372.0	0.0	2.331000e+03	6.27	6.00	2.10	4.400000e+00	2.00	3.00	4.00	4.00	5.00	6.00	7.00	8.
V-8	372.0	0.0	4.047800e+05	1088.12	805.00	995.83	9.916698e+05	40.00	97.10	170.00	220.00	440.00	805.00	1300.00	2300.
V-11	372.0	0.0	1.566491e+06	4211.00	3629.00	1776.65	3.156468e+06	1562.00	1580.00	2028.00	2264.00	2841.75	3629.00	6024.25	6790.
V-12	372.0	0.0	3.512720e+04	94.43	74.90	62.89	3.955330e+03	12.10	12.53	20.72	29.64	45.60	74.90	137.40	202.
V-13	372.0	0.0	3.275478e+04	88.05	79.28	49.36	2.436830e+03	10.03	11.12	23.12	30.08	51.63	79.28	125.83	161.
V-14	372.0	0.0	1.341180e+03	3.61	3.25	1.62	2.610000e+00	0.92	0.92	1.34	1.72	2.47	3.25	4.72	6.
V-15	372.0	0.0	2.384935e+08	641111.64	445458.35	542163.77	2.939415e+11	38193.64	40197.17	67670.67	92923.07	183726.00	445458.35	1059966.20	1612714.
V-16	372.0	0.0	1.787666e+06	4805.55	3819.00	3947.16	1.558004e+07	287.20	324.40	643.28	1163.30	1979.00	3819.00	6622.50	10855.
V-17	372.0	0.0	3.670815e+04	98.68	87.05	73.02	5.331250e+03	13.60	14.27	21.63	25.89	39.70	87.05	117.40	227.
V-18	372.0	0.0	6.770479e+04	182.00	162.75	110.71	1.225701e+04	17.03	23.99	52.20	60.35	93.00	162.75	242.27	334.
V-19	372.0	0.0	7.016423e+06	18861.35	10445.60	21313.73	4.542752e+08	154.40	220.38	732.40	1622.28	3622.15	10445.60	21723.40	54857.

Figure 4: Statistical Summaries

The Y-variable was transformed using log transformation for the OLS method.

[32]:
```
sns.distplot(data['V-10'])
plt.show()
```



[34]:
```
data['ln_V-10'] = np.log(data['V-10'])
```

[36]:
```
sns.distplot(data['ln_V-10'])
plt.show()
```

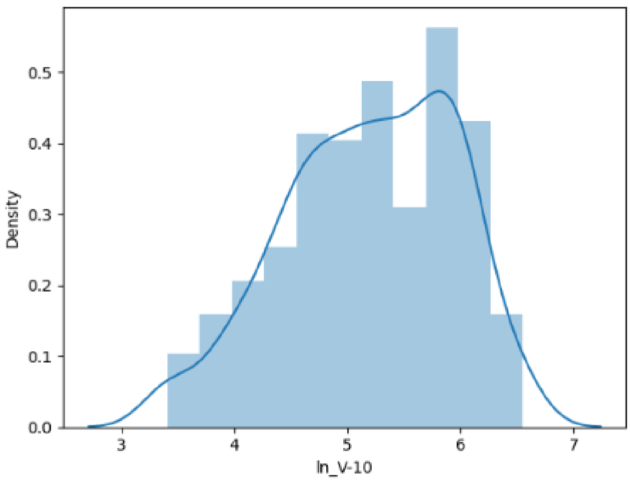


Figure 5: Y-Variable Distribution

8 Model Building – Statistical Approach

The first model built in the research was the OLS model.

```
[69]: print(lm0.summary2())
```

```
                 Results: Ordinary least squares
==================================================================
Model:              OLS              Adj. R-squared:     0.979
Dependent Variable: ln_V_10          AIC:                -388.7449
Date:               2024-08-10 21:28 BIC:                -281.9244
No. Observations:   260              Log-Likelihood:     224.37
Df Model:           29               F-statistic:        420.8
Df Residuals:       230              Prob (F-statistic): 2.69e-182
R-squared:          0.981            Scale:              0.011782
-------------------------------------------------------------------
                Coef.   Std.Err.    t     P>|t|    [0.025   0.975]
-------------------------------------------------------------------
Intercept       1.9949   0.2447   8.1537 0.0000   1.5128   2.4770
V_1            -0.0061   0.0024  -2.5637 0.0110  -0.0108  -0.0014
V_10            0.0021   0.0003   6.8257 0.0000   0.0015   0.0028
V_11            0.0000   0.0000   1.2163 0.2251  -0.0000   0.0000
V_12           -0.0023   0.0034  -0.6684 0.5046  -0.0090   0.0045
V_13           -0.0010   0.0026  -0.3636 0.7165  -0.0061   0.0042
V_14           -0.0094   0.0136  -0.6942 0.4883  -0.0361   0.0173
V_15            0.0000   0.0000   3.7724 0.0002   0.0000   0.0000
V_16            0.0000   0.0000   1.8339 0.0680  -0.0000   0.0001
V_17           -0.0027   0.0011  -2.5095 0.0128  -0.0048  -0.0006
V_18           -0.0003   0.0003  -1.1901 0.2352  -0.0008   0.0002
V_19           -0.0000   0.0000  -1.7079 0.0890  -0.0000   0.0000
V_2             0.0000   0.0000   1.6234 0.1059  -0.0000   0.0001
V_20            0.0475   0.0166   2.8624 0.0046   0.0148   0.0801
V_21           -0.0004   0.0001  -3.6750 0.0003  -0.0006  -0.0002
V_22            0.0002   0.0001   1.4301 0.1540  -0.0001   0.0004
V_23            0.0000   0.0000   2.3903 0.0176   0.0000   0.0000
V_24            0.0000   0.0000   1.3643 0.1738  -0.0000   0.0001
V_25            0.0250   0.0083   2.9945 0.0030   0.0085   0.0414
V_26           -0.0096   0.0061  -1.5697 0.1179  -0.0217   0.0025
V_27            0.0000   0.0000   1.1458 0.2531  -0.0000   0.0000
V_28            0.0000   0.0000   2.3587 0.0192   0.0000   0.0000
V_29           -0.0000   0.0000  -1.3973 0.1637  -0.0000   0.0000
V_3            -0.0001   0.0001  -1.4355 0.1525  -0.0002   0.0000
V_4            -0.0001   0.0000  -1.9765 0.0493  -0.0002  -0.0000
V_5            -0.0021   0.0005  -4.5185 0.0000  -0.0030  -0.0012
V_6             0.0015   0.0001  17.2774 0.0000   0.0013   0.0017
V_7             0.0269   0.0051   5.2414 0.0000   0.0168   0.0370
V_8            -0.0001   0.0001  -0.8861 0.3765  -0.0002   0.0001
V_9             0.0000   0.0000   0.1838 0.8544  -0.0001   0.0001
------------------------------------------------------------------
Omnibus:              38.540       Durbin-Watson:          2.016
Prob(Omnibus):        0.000        Jarque-Bera (JB):       309.417
Skew:                 0.076        Prob(JB):               0.000
Kurtosis:             8.342        Condition No.:          53385943
==================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
[2] The condition number is large, 5.34e+07. This might indicate
that there are strong multicollinearity or other numerical
problems.
```

Figure 6: Summary of OLS Method

As the results were not satisfactory, a new model was built using fewer attributes.

```
[78]: lm1 = smf.ols( formula = model_param_1, data = train ).fit()
```

```
[80]: print(lm1.summary2())
```

```
                 Results: Ordinary least squares
==================================================================
Model:              OLS              Adj. R-squared:     0.975
Dependent Variable: ln_V_10          AIC:                -359.2639
Date:               2024-08-10 21:28 BIC:                -312.9751
No. Observations:   260              Log-Likelihood:     192.63
Df Model:           12               F-statistic:        851.0
Df Residuals:       247              Prob (F-statistic): 2.97e-193
R-squared:          0.976            Scale:              0.014005
-------------------------------------------------------------------
                Coef.  Std.Err.    t     P>|t|   [0.025   0.975]
-------------------------------------------------------------------
Intercept       2.0296   0.1639 12.3802 0.0000   1.7067   2.3525
V_1            -0.0028   0.0021 -1.3735 0.1708  -0.0069   0.0012
V_10            0.0025   0.0003  8.5067 0.0000   0.0019   0.0030
V_15            0.0000   0.0000  1.4241 0.1557  -0.0000   0.0000
V_20            0.0478   0.0110  4.3468 0.0000   0.0261   0.0695
V_21           -0.0003   0.0001 -4.5898 0.0000  -0.0005  -0.0002
V_23            0.0000   0.0000  3.5909 0.0004   0.0000   0.0000
V_25            0.0184   0.0013 14.4341 0.0000   0.0159   0.0210
V_28            0.0000   0.0000  0.3218 0.7479  -0.0000   0.0000
V_4            -0.0001   0.0000 -1.9547 0.0517  -0.0001   0.0000
V_5            -0.0031   0.0004 -8.3108 0.0000  -0.0038  -0.0023
V_6             0.0016   0.0001 19.5324 0.0000   0.0015   0.0018
V_7             0.0245   0.0051  4.7588 0.0000   0.0143   0.0346
------------------------------------------------------------------
Omnibus:              31.271       Durbin-Watson:          2.062
Prob(Omnibus):        0.000        Jarque-Bera (JB):       105.989
Skew:                 -0.428       Prob(JB):               0.000
Kurtosis:             6.008        Condition No.:          18598948
==================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
[2] The condition number is large, 1.86e+07. This might indicate
that there are strong multicollinearity or other numerical
problems.
```

Figure 7: Summary of OLS Method-2

Predicting The values

```
[83]: # predict the values on training and testing
train_predict_lr = np.exp(lm1.predict(train))
test_predict_lr = np.exp(lm1.predict(test))
```

Evaluating the model

```
[86]: MSE_train_ols = mean_squared_error(train.V_10,train_predict_lr).round(2)
MSE_test_ols = mean_squared_error(test.V_10,test_predict_lr).round(2)

RMSE_train_ols = np.sqrt(MSE_train_ols).round(2)
RMSE_test_ols = np.sqrt(MSE_test_ols).round(2)

# print the values of MAPE for train and test
print('MSE of training data: ', MSE_train_ols,  ' | ', 'MSE of testing data: ', MSE_test_ols)

# print the values of MAPE for train and test
print('RMSE of training data: ', RMSE_train_ols,  ' | ', 'RMSE of testing data: ', RMSE_test_ols)

MSE of training data:  1387.54  |  MSE of testing data:  1923.91
RMSE of training data:  37.25  |  RMSE of testing data:  43.86
```

```
[88]: MAE_train_ols = mean_absolute_error(train.V_10,train_predict_lr).round(2)
MAE_test_ols = mean_absolute_error(test.V_10,test_predict_lr).round(2)

# print the values of MAPE for train and test
print('MAE of training data: ', MAE_train_ols,  ' | ', 'MAE of testing data: ', MAE_test_ols)

MAE of training data:  18.05  |  MAE of testing data:  24.2
```

Figure 8: Evaluation of the OLS Method

9 Model Building – ML Based Approach

The next phase of the study uses machine learning models on the same dataset for a comparative analysis. As there are insignificant variables, the first RFE was done.

RFE

```
[96]: X = data[data.columns.difference(['V_10'])]
classifier = RandomForestClassifier()
rfe = RFE(classifier,n_features_to_select=15)
rfe = rfe.fit(X, data[['V_10']] )
rfe_feat = list(X.columns[rfe.support_])
list(rfe_feat)

[96]: ['V_1',
 'V_17',
 'V_18',
 'V_19',
 'V_2',
 'V_22',
 'V_26',
 'V_27',
 'V_3',
 'V_4',
 'V_5',
 'V_6',
 'V_7',
 'V_8',
 'V_9']

[97]: X = X[rfe_feat]

[98]: X.shape

[98]: (372, 15)

[99]: y = pd.DataFrame(data.V_10)

[100]: #Splitting the data for sklearn methods
train_y, test_y, train_X, test_X = train_test_split(y,X, test_size=0.3, random_state=123)
```

Figure 9: RFE

Random Forest

```
[117]: param_grid = {
           'n_estimators': [100, 200, 300],
           'max_depth': [3, 5, 10,],
           'min_samples_split': [2, 5, 10],
           'min_samples_leaf': [1, 2, 4],
       }
```

```
[119]: RF = GridSearchCV( estimator = RandomForestRegressor( random_state = 20 ),
                         param_grid = param_grid, cv = 5,
                             scoring = 'neg_mean_squared_error',
                                 n_jobs = -1,
                                     verbose = True)
       RF.fit( train_X, train_y )
```

```
       Fitting 5 folds for each of 81 candidates, totalling 405 fits
[119]: ▸ GridSearchCV
       ▸ estimator: RandomForestRegressor
           ▸ RandomForestRegressor
```

```
[120]: RF.best_params_
```

```
[120]: {'max_depth': 10,
        'min_samples_leaf': 1,
        'min_samples_split': 2,
        'n_estimators': 100}
```

```
[121]: RF.best_score_
```

```
[121]: -1171.9558546592164
```

```
[122]: train_pred_rf = RF.predict(train_X)
       test_pred_rf = RF.predict(test_X)
```

Figure 10: Random Forest

Evaluation

```
124]: MSE_train_rf = mean_squared_error(train_y,train_pred_rf).round(2)
      MSE_test_rf = mean_squared_error(test_y,test_pred_rf).round(2)

      RMSE_train_rf = np.sqrt(MSE_train_rf).round(2)
      RMSE_test_rf = np.sqrt(MSE_test_rf).round(2)

      # print the values of MAPE for train and test
      print('MSE of training data: ', MSE_train_rf,  ' | ', 'MSE of testing data: ', MSE_test_rf)

      # print the values of MAPE for train and test
      print('RMSE of training data: ', RMSE_train_rf,  ' | ', 'RMSE of testing data: ', RMSE_test_rf)
```

```
      MSE of training data:  139.92  |  MSE of testing data:  709.93
      RMSE of training data:  11.83  |  RMSE of testing data:  26.64
```

```
130]: MAE_train_rf = mean_absolute_error(train_y,train_pred_rf).round(2)
      MAE_test_rf = mean_absolute_error(test_y,test_pred_rf).round(2)

      # print the values of MAPE for train and test
      print('MAE of training data: ', MAE_train_rf,  ' | ', 'MAE of testing data: ', MAE_test_rf)
```

```
      MAE of training data:  7.68  |  MAE of testing data:  18.87
```

Figure 11: Evaluation of Random Forest

One ANN model was also built in the research for the comparative analysis of machine learning with deep learning.

ANN

```
[133]: sc = StandardScaler()
       std_data = sc.fit_transform(train_X)

[135]: std_data_train = pd.DataFrame(std_data, columns=train_X.columns, index = train_X.index )
       std_data_train.shape

[135]: (260, 15)

[137]: std_data_test = pd.DataFrame(sc.transform(test_X), columns=test_X.columns, index = test_X.index)
       std_data_test.shape

[137]: (112, 15)

[139]: # Initialize the ANN model
       ANN = Sequential()
       ANN.add(Dense(64, input_dim= 15, activation='relu'))
       ANN.add(Dense(32, activation='relu'))
       ANN.add(Dense(16, activation='relu'))
       ANN.add(Dense(1))  # Output layer

[141]: ANN.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	1,024
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 1)	17

Total params: 3,649 (14.25 KB)

Trainable params: 3,649 (14.25 KB)

Non-trainable params: 0 (0.00 B)

Figure 12: ANN

Evaluation

```
[150]: MSE_train_ANN = mean_squared_error(train_y,np.round(abs(train_pred_ann))).round(2)
       MSE_test_ANN = mean_squared_error(test_y,np.round(abs(test_pred_ann))).round(2)

       RMSE_train_ANN = np.sqrt(MSE_train_ANN).round(2)
       RMSE_test_ANN = np.sqrt(MSE_test_ANN).round(2)

       # print the values of MAPE for train and test
       print('MSE of training data: ', MSE_train_ANN,  ' | ', 'MSE of testing data: ', MSE_test_ANN)

       # print the values of MAPE for train and test
       print('RMSE of training data: ', RMSE_train_ANN,  ' | ', 'RMSE of testing data: ', RMSE_test_ANN)

       MSE of training data:  1295.05  |  MSE of testing data:  1690.45
       RMSE of training data:  35.99  |  RMSE of testing data:  41.12

[151]: MAE_train_ANN = mean_absolute_error(train_y,np.round(abs(train_pred_ann))).round(2)
       MAE_test_ANN = mean_absolute_error(test_y,np.round(abs(test_pred_ann))).round(2)

       # print the values of MAPE for train and test
       print('MAE of training data: ', MAE_train_ANN,  ' | ', 'MAE of testing data: ', MAE_test_ANN)

       MAE of training data:  28.54  |  MAE of testing data:  32.27
```

Figure 13: Evaluation of ANN

10 XAI

The study has incorporated SHAP for the Explainability of features

XAI

```
[97]: explainer = shap.TreeExplainer(RF.best_estimator_)
```

```
[98]: shap_values = explainer(test_X)
      np.shape(shap_values.values)
```

```
[98]: (112, 15)
```

```
[99]: shap.plots.waterfall(shap_values[0])
```

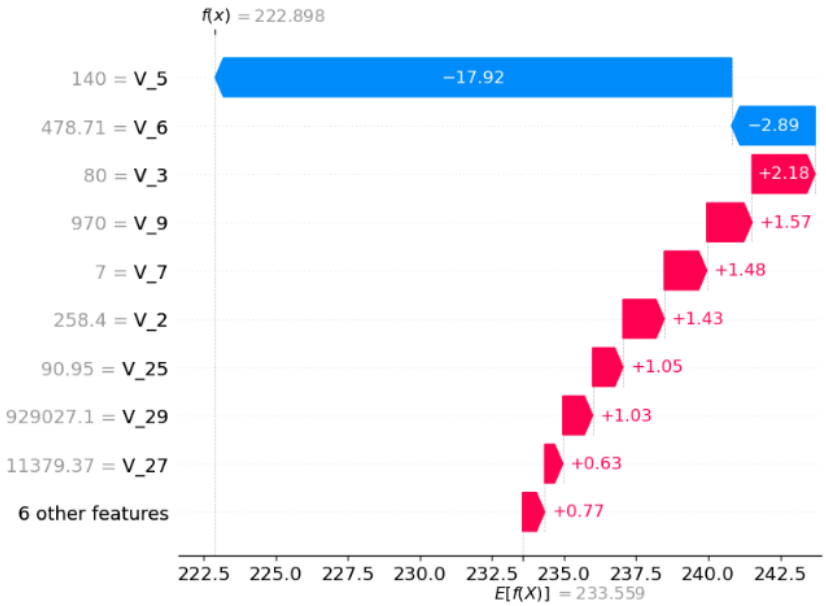


Figure 14: SHAP Values of Features

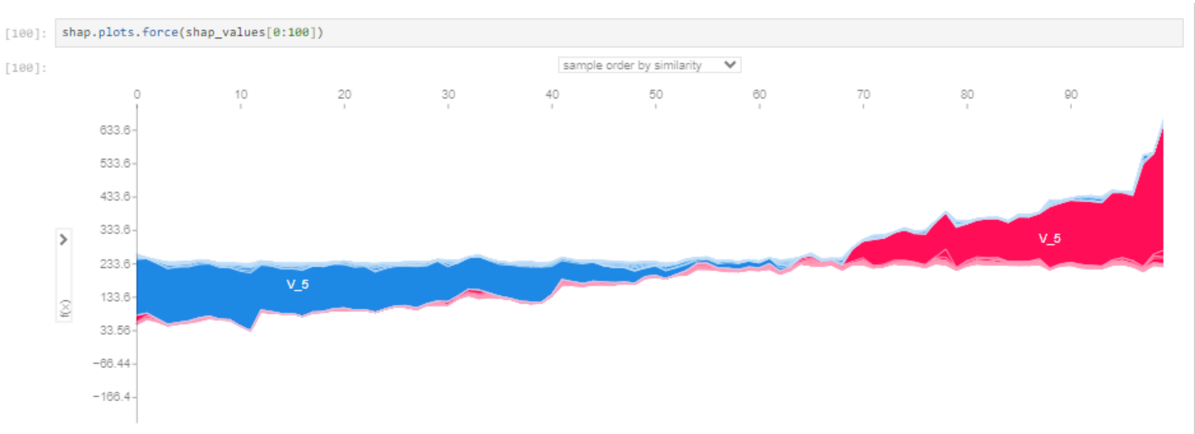


Figure 15: SHAP Summary Plot

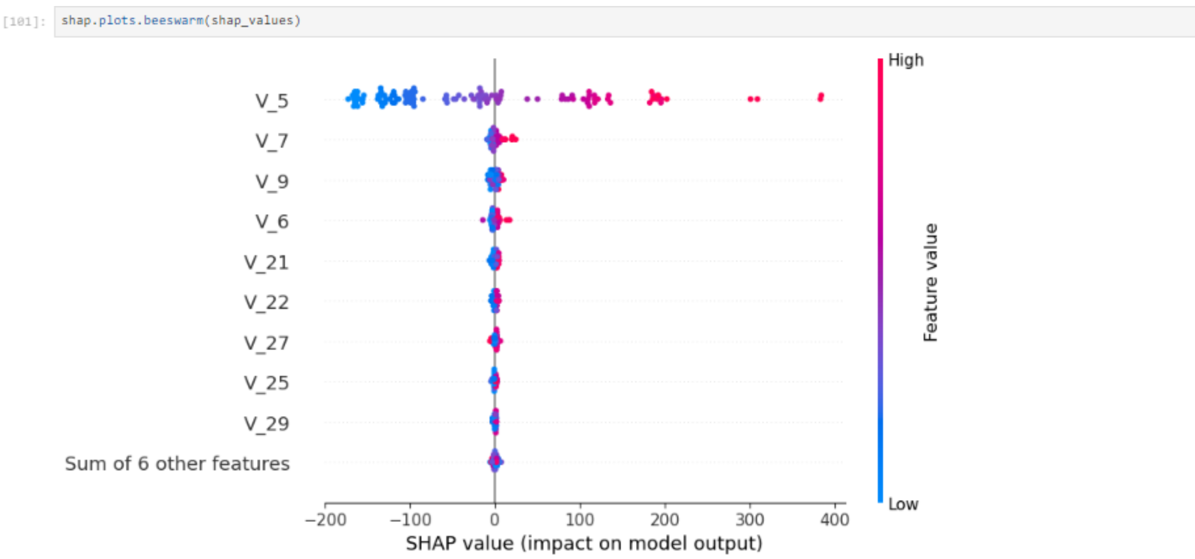


Figure 16: SHAP Summary Dot- Plot