Final Project Part 3

Table of Contents

Inference

In [1]:

```
import numpy as np
import pandas as pd
import xgboost as xgb
import statsmodels.api as sm
import sklearn.linear_model as lm
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR


import seaborn as sns
sns.set(rc = {'axes.titlesize': 24,
             'axes.labelsize': 20,
             'xtick.labelsize': 12,
             'ytick.labelsize': 12,
             'figure.figsize': (12, 6)})
```

In [2]:

```
sic = pd.read_pickle('sic_fp.pkl')
sic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 24676 entries, (2011, 1958) to (3999, 2011)
Data columns (total 9 columns):
l_vadd      24676 non-null float64
l_emp       24676 non-null float64
l_invest    24676 non-null float64
l_pay       24676 non-null float64
```

```
l_matcost    24676 non-null float64
l_vship      24676 non-null float64
l_cap        24676 non-null float64
l_invent     24676 non-null float64
l_energy     24676 non-null float64
dtypes: float64(9)
memory usage: 1.8 MB
```

In [3]:

```
y = sic['l_vadd']
x = sic.drop(columns = 'l_vadd')

y_train, y_test = train_test_split(y, train_size = 1/20, random_state = 490)
x_train, x_test = train_test_split(x, train_size = 1/20, random_state = 490)

ss = StandardScaler()
x_train_std = pd.DataFrame(ss.fit(x_train).transform(x_train),
                          columns = x_train.columns,
                          index = x_train.index)
x_test_std = pd.DataFrame(ss.fit(x_test).transform(x_test),
                         columns = x_test.columns,
                         index = x_test.index)

x_train_std_c     = sm.add_constant(x_train_std)
x_test_std_c      = sm.add_constant(x_test_std)
x_train_c    = sm.add_constant(x_train)
x_test_c     = sm.add_constant(x_test)
```

```
C:\Users\tanse\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [4]:

```
param_grid = [
    {'alpha': 10**np.linspace(-6, -4, num = 10)}
]


cv_lasso = lm.Lasso(fit_intercept = False, normalize = False,
                    random_state = 490)
grid_search = GridSearchCV(cv_lasso, param_grid, cv = 5,
                         scoring = 'neg_root_mean_squared_error')
grid_search.fit(x_train_std_c, y_train)
best = grid_search.best_params_['alpha']
best
```

Out[4]: 2.1544346900318823e-05

In [5]:
```
fit_lasso_tuned = sm.OLS(y_train, x_train_std_c).fit_regularized(alpha = best)
beta = fit_lasso_tuned.params #fitting on non regularized standardized model
beta.index[beta == 0]
x_train_trim = x_train_std_c.loc[:, ~x_train_std_c.columns.isin(beta.index[beta == 0])]
x_test_trim = x_test_std_c.loc[:, ~x_test_std_c.columns.isin(beta.index[beta == 0])]
```

In [6]:
```
fit_std_final = sm.OLS(y_train, x_train_trim).fit() #testing on non-regularized values
fit_std_final.summary2()
```

Out[6]:

Model:	OLS	Adj. R-squared:	0.993
Dependent Variable:	l_vadd	AIC:	-1815.8247
Date:	2021-04-19 20:22	BIC:	-1769.7698
No. Observations:	1233	Log-Likelihood:	916.91
Df Model:	8	F-statistic:	2.251e+04
Df Residuals:	1224	Prob (F-statistic):	0.00
R-squared:	0.993	Scale:	0.013329

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	6.7953	0.0033	2066.7960	0.0000	6.7889	6.8018
l_emp	-0.0291	0.0071	-4.1069	0.0000	-0.0430	-0.0152
l_invest	0.0802	0.0120	6.6726	0.0000	0.0566	0.1038
l_pay	0.2384	0.0156	15.3025	0.0000	0.2079	0.2690
l_matcost	-1.2348	0.0247	-50.0499	0.0000	-1.2832	-1.1864
l_vship	2.2908	0.0342	66.9692	0.0000	2.2237	2.3579
l_cap	-0.0403	0.0098	-4.1066	0.0000	-0.0595	-0.0210
l_invent	0.0582	0.0095	6.0911	0.0000	0.0394	0.0769
l_energy	-0.0031	0.0091	-0.3382	0.7353	-0.0210	0.0148

Omnibus:	668.953	Durbin-Watson:	1.964
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7106.942
Skew:	-2.297	Prob(JB):	0.000
Kurtosis:	13.827	Condition No.:	34

Top three most significant variables are l_vship, l_matcost, and l_pay, each with extremely high t statistics.

- l_vship: A one percent increase in the total value of shipments is associated with a decrease of total value added by 2.2908 percent.
- l_matcost: A one percent increase in the total cost of materials is associated with a decrease of total value added by 1.2348 percent.
- l_pay: A one percent increase in the total payroll is associated with an increase of total value added by 0.2384 percent.

In [7]:

```
rmse_ols = np.sqrt(np.mean((y_test - fit_std_final.predict(x_test_trim))**2))
rmse_ols
```

Out[7]: 0.12736127258697186

Prediction

TOP

SVM

In [8]:

```
param_grid = {
    'degree': [1,2,3,4],
    'C': 10. ** np.arange(-2,2),
    'epsilon': 10. ** np.arange(-2,2)
}
svr_cv = SVR(kernel = 'poly')
grid_search = GridSearchCV(svr_cv, param_grid, cv = 5,
                           scoring = 'neg_root_mean_squared_error')
grid_search.fit(x_train_std, y_train)
best = grid_search.best_params_
best
```

Out[8]: {'C': 10.0, 'degree': 1, 'epsilon': 0.1}

In [9]:
```
svr_poly = SVR(kernel = 'poly', degree = best['degree'],
              C = best['C'], epsilon = best['epsilon'])
svr_poly.fit(x_train_std, y_train)

rmse_poly = np.sqrt(np.mean((y_test - svr_poly.predict(x_test_std))**2))
rmse_poly
```

Out[9]:
```
0.12668720173565048
```

XGBoost

In [10]:
```
x_train_train, x_train_test, y_train_train, y_train_test = train_test_split(x_train_std, y_train,
                                                                             train_size = 1/2,
                                                                            random_state = 490)
```

In [11]:
```
clf_xgb = xgb.XGBRegressor(n_estimators = 500, max_depth = 6, learning_rate = 0.1,
                          random_state = 490, use_label_encoder = False)
clf_xgb.fit(x_train_train, y_train_train, eval_set = [(x_train_test, y_train_test)],
           early_stopping_rounds = 5)
```

```
[0]	validation_0-rmse:5.84672
[1]	validation_0-rmse:5.27003
[2]	validation_0-rmse:4.74926
[3]	validation_0-rmse:4.28063
[4]	validation_0-rmse:3.85996
[5]	validation_0-rmse:3.48322
[6]	validation_0-rmse:3.14207
[7]	validation_0-rmse:2.83538
[8]	validation_0-rmse:2.55781
[9]	validation_0-rmse:2.31080
[10]	validation_0-rmse:2.08797
[11]	validation_0-rmse:1.88786
[12]	validation_0-rmse:1.70642
[13]	validation_0-rmse:1.54234
[14]	validation_0-rmse:1.39549
[15]	validation_0-rmse:1.26459
[16]	validation_0-rmse:1.14680
[17]	validation_0-rmse:1.04145
[18]	validation_0-rmse:0.94652
[19]	validation_0-rmse:0.86214
[20]	validation_0-rmse:0.78587
```

```
[21]	validation_0-rmse:0.71766
[22]	validation_0-rmse:0.65713
[23]	validation_0-rmse:0.60376
[24]	validation_0-rmse:0.55631
[25]	validation_0-rmse:0.51372
[26]	validation_0-rmse:0.47519
[27]	validation_0-rmse:0.44173
[28]	validation_0-rmse:0.41300
[29]	validation_0-rmse:0.38603
[30]	validation_0-rmse:0.36245
[31]	validation_0-rmse:0.34257
[32]	validation_0-rmse:0.32513
[33]	validation_0-rmse:0.30917
[34]	validation_0-rmse:0.29572
[35]	validation_0-rmse:0.28326
[36]	validation_0-rmse:0.27332
[37]	validation_0-rmse:0.26505
[38]	validation_0-rmse:0.25751
[39]	validation_0-rmse:0.25140
[40]	validation_0-rmse:0.24527
[41]	validation_0-rmse:0.24069
[42]	validation_0-rmse:0.23655
[43]	validation_0-rmse:0.23240
[44]	validation_0-rmse:0.22908
[45]	validation_0-rmse:0.22607
[46]	validation_0-rmse:0.22357
[47]	validation_0-rmse:0.22126
[48]	validation_0-rmse:0.21936
[49]	validation_0-rmse:0.21788
[50]	validation_0-rmse:0.21655
[51]	validation_0-rmse:0.21498
[52]	validation_0-rmse:0.21386
[53]	validation_0-rmse:0.21266
[54]	validation_0-rmse:0.21173
[55]	validation_0-rmse:0.21068
[56]	validation_0-rmse:0.20934
[57]	validation_0-rmse:0.20886
[58]	validation_0-rmse:0.20818
[59]	validation_0-rmse:0.20717
[60]	validation_0-rmse:0.20633
[61]	validation_0-rmse:0.20546
[62]	validation_0-rmse:0.20450
[63]	validation_0-rmse:0.20428
[64]	validation_0-rmse:0.20345
[65]	validation_0-rmse:0.20277
[66]	validation_0-rmse:0.20229
[67]	validation_0-rmse:0.20229
```

```
[68]	validation_0-rmse:0.20190
[69]	validation_0-rmse:0.20122
[70]	validation_0-rmse:0.20119
[71]	validation_0-rmse:0.20096
[72]	validation_0-rmse:0.20034
[73]	validation_0-rmse:0.20031
[74]	validation_0-rmse:0.19989
[75]	validation_0-rmse:0.19961
[76]	validation_0-rmse:0.19959
[77]	validation_0-rmse:0.19953
[78]	validation_0-rmse:0.19897
[79]	validation_0-rmse:0.19890
[80]	validation_0-rmse:0.19879
[81]	validation_0-rmse:0.19869
[82]	validation_0-rmse:0.19823
[83]	validation_0-rmse:0.19783
[84]	validation_0-rmse:0.19773
[85]	validation_0-rmse:0.19739
[86]	validation_0-rmse:0.19734
[87]	validation_0-rmse:0.19689
[88]	validation_0-rmse:0.19690
[89]	validation_0-rmse:0.19680
[90]	validation_0-rmse:0.19660
[91]	validation_0-rmse:0.19653
[92]	validation_0-rmse:0.19644
[93]	validation_0-rmse:0.19611
[94]	validation_0-rmse:0.19597
[95]	validation_0-rmse:0.19596
[96]	validation_0-rmse:0.19598
[97]	validation_0-rmse:0.19582
[98]	validation_0-rmse:0.19565
[99]	validation_0-rmse:0.19557
[100]	validation_0-rmse:0.19544
[101]	validation_0-rmse:0.19520
[102]	validation_0-rmse:0.19511
[103]	validation_0-rmse:0.19498
[104]	validation_0-rmse:0.19490
[105]	validation_0-rmse:0.19484
[106]	validation_0-rmse:0.19481
[107]	validation_0-rmse:0.19464
[108]	validation_0-rmse:0.19460
[109]	validation_0-rmse:0.19428
[110]	validation_0-rmse:0.19425
[111]	validation_0-rmse:0.19421
[112]	validation_0-rmse:0.19411
[113]	validation_0-rmse:0.19411
[114]	validation_0-rmse:0.19397
```

```
[115]	validation_0-rmse:0.19400
[116]	validation_0-rmse:0.19376
[117]	validation_0-rmse:0.19368
[118]	validation_0-rmse:0.19358
[119]	validation_0-rmse:0.19358
[120]	validation_0-rmse:0.19360
[121]	validation_0-rmse:0.19360
[122]	validation_0-rmse:0.19351
[123]	validation_0-rmse:0.19329
[124]	validation_0-rmse:0.19334
[125]	validation_0-rmse:0.19330
[126]	validation_0-rmse:0.19324
[127]	validation_0-rmse:0.19301
[128]	validation_0-rmse:0.19302
[129]	validation_0-rmse:0.19297
[130]	validation_0-rmse:0.19291
[131]	validation_0-rmse:0.19286
[132]	validation_0-rmse:0.19285
[133]	validation_0-rmse:0.19280
[134]	validation_0-rmse:0.19275
[135]	validation_0-rmse:0.19272
[136]	validation_0-rmse:0.19268
[137]	validation_0-rmse:0.19265
[138]	validation_0-rmse:0.19264
[139]	validation_0-rmse:0.19265
[140]	validation_0-rmse:0.19261
[141]	validation_0-rmse:0.19259
[142]	validation_0-rmse:0.19258
[143]	validation_0-rmse:0.19248
[144]	validation_0-rmse:0.19233
[145]	validation_0-rmse:0.19225
[146]	validation_0-rmse:0.19225
[147]	validation_0-rmse:0.19220
[148]	validation_0-rmse:0.19201
[149]	validation_0-rmse:0.19194
[150]	validation_0-rmse:0.19194
[151]	validation_0-rmse:0.19198
[152]	validation_0-rmse:0.19197
[153]	validation_0-rmse:0.19192
[154]	validation_0-rmse:0.19190
[155]	validation_0-rmse:0.19190
[156]	validation_0-rmse:0.19189
[157]	validation_0-rmse:0.19184
[158]	validation_0-rmse:0.19183
[159]	validation_0-rmse:0.19180
[160]	validation_0-rmse:0.19172
[161]	validation_0-rmse:0.19170
```

```
[162]	validation_0-rmse:0.19166
[163]	validation_0-rmse:0.19157
[164]	validation_0-rmse:0.19151
[165]	validation_0-rmse:0.19151
[166]	validation_0-rmse:0.19151
[167]	validation_0-rmse:0.19143
[168]	validation_0-rmse:0.19142
[169]	validation_0-rmse:0.19141
[170]	validation_0-rmse:0.19136
[171]	validation_0-rmse:0.19131
[172]	validation_0-rmse:0.19130
[173]	validation_0-rmse:0.19125
[174]	validation_0-rmse:0.19119
[175]	validation_0-rmse:0.19120
[176]	validation_0-rmse:0.19112
[177]	validation_0-rmse:0.19110
[178]	validation_0-rmse:0.19110
[179]	validation_0-rmse:0.19103
[180]	validation_0-rmse:0.19101
[181]	validation_0-rmse:0.19101
[182]	validation_0-rmse:0.19098
[183]	validation_0-rmse:0.19084
[184]	validation_0-rmse:0.19081
[185]	validation_0-rmse:0.19076
[186]	validation_0-rmse:0.19073
[187]	validation_0-rmse:0.19060
[188]	validation_0-rmse:0.19056
[189]	validation_0-rmse:0.19051
[190]	validation_0-rmse:0.19051
[191]	validation_0-rmse:0.19048
[192]	validation_0-rmse:0.19039
[193]	validation_0-rmse:0.19030
[194]	validation_0-rmse:0.19031
[195]	validation_0-rmse:0.19030
[196]	validation_0-rmse:0.19031
[197]	validation_0-rmse:0.19030
[198]	validation_0-rmse:0.19028
[199]	validation_0-rmse:0.19028
[200]	validation_0-rmse:0.19026
[201]	validation_0-rmse:0.19023
[202]	validation_0-rmse:0.19022
[203]	validation_0-rmse:0.19016
[204]	validation_0-rmse:0.19017
[205]	validation_0-rmse:0.19015
[206]	validation_0-rmse:0.19014
[207]	validation_0-rmse:0.19012
[208]	validation_0-rmse:0.19011
```

```
[209]	validation_0-rmse:0.19009
[210]	validation_0-rmse:0.19006
[211]	validation_0-rmse:0.19005
[212]	validation_0-rmse:0.19002
[213]	validation_0-rmse:0.19001
[214]	validation_0-rmse:0.18999
[215]	validation_0-rmse:0.18991
[216]	validation_0-rmse:0.18991
[217]	validation_0-rmse:0.18991
[218]	validation_0-rmse:0.18992
[219]	validation_0-rmse:0.18993
[220]	validation_0-rmse:0.18985
[221]	validation_0-rmse:0.18985
[222]	validation_0-rmse:0.18984
[223]	validation_0-rmse:0.18982
[224]	validation_0-rmse:0.18982
[225]	validation_0-rmse:0.18981
[226]	validation_0-rmse:0.18981
[227]	validation_0-rmse:0.18977
[228]	validation_0-rmse:0.18976
[229]	validation_0-rmse:0.18976
[230]	validation_0-rmse:0.18976
[231]	validation_0-rmse:0.18972
[232]	validation_0-rmse:0.18969
[233]	validation_0-rmse:0.18969
[234]	validation_0-rmse:0.18964
[235]	validation_0-rmse:0.18965
[236]	validation_0-rmse:0.18959
[237]	validation_0-rmse:0.18958
[238]	validation_0-rmse:0.18957
[239]	validation_0-rmse:0.18956
[240]	validation_0-rmse:0.18953
[241]	validation_0-rmse:0.18952
[242]	validation_0-rmse:0.18950
[243]	validation_0-rmse:0.18949
[244]	validation_0-rmse:0.18947
[245]	validation_0-rmse:0.18945
[246]	validation_0-rmse:0.18943
[247]	validation_0-rmse:0.18942
[248]	validation_0-rmse:0.18942
[249]	validation_0-rmse:0.18941
[250]	validation_0-rmse:0.18942
[251]	validation_0-rmse:0.18941
[252]	validation_0-rmse:0.18939
[253]	validation_0-rmse:0.18936
[254]	validation_0-rmse:0.18935
[255]	validation_0-rmse:0.18936
```

```
[256]	validation_0-rmse:0.18935
[257]	validation_0-rmse:0.18935
[258]	validation_0-rmse:0.18933
[259]	validation_0-rmse:0.18930
[260]	validation_0-rmse:0.18930
[261]	validation_0-rmse:0.18930
[262]	validation_0-rmse:0.18929
[263]	validation_0-rmse:0.18928
[264]	validation_0-rmse:0.18928
[265]	validation_0-rmse:0.18927
[266]	validation_0-rmse:0.18926
[267]	validation_0-rmse:0.18924
[268]	validation_0-rmse:0.18923
[269]	validation_0-rmse:0.18923
[270]	validation_0-rmse:0.18919
[271]	validation_0-rmse:0.18917
[272]	validation_0-rmse:0.18914
[273]	validation_0-rmse:0.18914
[274]	validation_0-rmse:0.18913
[275]	validation_0-rmse:0.18912
[276]	validation_0-rmse:0.18911
[277]	validation_0-rmse:0.18911
[278]	validation_0-rmse:0.18911
[279]	validation_0-rmse:0.18911
[280]	validation_0-rmse:0.18910
[281]	validation_0-rmse:0.18910
[282]	validation_0-rmse:0.18910
[283]	validation_0-rmse:0.18909
[284]	validation_0-rmse:0.18909
[285]	validation_0-rmse:0.18909
[286]	validation_0-rmse:0.18909
[287]	validation_0-rmse:0.18909
[288]	validation_0-rmse:0.18907
[289]	validation_0-rmse:0.18907
[290]	validation_0-rmse:0.18905
[291]	validation_0-rmse:0.18906
[292]	validation_0-rmse:0.18906
[293]	validation_0-rmse:0.18903
[294]	validation_0-rmse:0.18903
[295]	validation_0-rmse:0.18903
[296]	validation_0-rmse:0.18903
[297]	validation_0-rmse:0.18903
[298]	validation_0-rmse:0.18902
[299]	validation_0-rmse:0.18900
[300]	validation_0-rmse:0.18900
[301]	validation_0-rmse:0.18900
[302]	validation_0-rmse:0.18899
```

```
[303]	validation_0-rmse:0.18899
[304]	validation_0-rmse:0.18898
[305]	validation_0-rmse:0.18898
[306]	validation_0-rmse:0.18898
[307]	validation_0-rmse:0.18897
[308]	validation_0-rmse:0.18896
[309]	validation_0-rmse:0.18896
[310]	validation_0-rmse:0.18894
[311]	validation_0-rmse:0.18894
[312]	validation_0-rmse:0.18894
[313]	validation_0-rmse:0.18894
[314]	validation_0-rmse:0.18893
[315]	validation_0-rmse:0.18892
[316]	validation_0-rmse:0.18892
[317]	validation_0-rmse:0.18892
[318]	validation_0-rmse:0.18890
[319]	validation_0-rmse:0.18889
[320]	validation_0-rmse:0.18890
[321]	validation_0-rmse:0.18888
[322]	validation_0-rmse:0.18888
[323]	validation_0-rmse:0.18887
[324]	validation_0-rmse:0.18888
[325]	validation_0-rmse:0.18888
[326]	validation_0-rmse:0.18888
[327]	validation_0-rmse:0.18888
[328]	validation_0-rmse:0.18887
[329]	validation_0-rmse:0.18887
[330]	validation_0-rmse:0.18888
[331]	validation_0-rmse:0.18888
[332]	validation_0-rmse:0.18888
```

Out[11]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=500, n_jobs=12, num_parallel_tree=1, random_state=490,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', use_label_encoder=False,
             validate_parameters=1, verbosity=None)
```

In [12]:

```
xgb_n_est = clf_xgb.best_iteration
xgb_n_est
```

Out[12]:

```
328
```

In [13]:
```
clf_xgb = xgb.XGBRegressor(n_estimators = xgb_n_est, max_depth = 5,
                           learning_rate = 0.1, random_state = 490,
                           use_label_encoder = False)
clf_xgb.fit(x_train_std, y_train)
```

Out[13]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=5,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=328, n_jobs=12, num_parallel_tree=1, random_state=490,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', use_label_encoder=False,
             validate_parameters=1, verbosity=None)
```

In [14]:
```
rmse_xgb = np.sqrt(np.mean((y_test - clf_xgb.predict(x_test_std))**2))
rmse_xgb
```

Out[14]:
```
0.14921002612012657
```

Random Forest

In [15]:
```
clf_rf = RandomForestRegressor(n_estimators = 1000,
                           random_state = 490,
                           max_features = 'sqrt',
                           oob_score = True)
clf_rf.fit(x_train_std, y_train)
```

Out[15]:
```
RandomForestRegressor(max_features='sqrt', n_estimators=1000, oob_score=True,
                      random_state=490)
```

In [16]:
```
rmse_rf = np.sqrt(np.mean((y_test - clf_rf.predict(x_test_std))**2))
rmse_rf
```

Out[16]:
```
0.21613780320275638
```

Comparison

TOP

Model	RMSE
OLS	0.12736127258697186
Support Vector Regression	0.12668720173565048
Extreme Gradient Boosting	0.14921002612012657
Random Forest	0.215241474310346

When compared to random forest, gradient boosting develops itself in each model, instead of independently like random forest. Extreme Gradient Boosting can be more flexible, but can run into issues quickly with overfitting the data if the hyperparameters are improperly tuned. Similarly, if a support vector regression uses an improper kernel it will be less accurate than extreme gradient boosting. Again, extreme gradient boosting likely comes out ahead due to its adaptive nature throughout its processing. Support vector machines are much more sensitive to outliers, so in cases with high variation or outliers random forest would be a better choice over the svm’s.

OLS is the easiest to interpret as x and y are both able to be evaluated against the predicted values from the estimated coefficients. Random forests can be interpreted through the model's fitted important features, which indicate a closer relationship between the dependent variable and the feature in question. Extreme gradient boosting follows this interpretation, but tries to improve and change the weighting for each model off of the previous model.

For this data, the polynomial support vector machine worked the best, having the lowest rmse over any of the models fitted.