



+ Code + Text

✓ RAM Disk

Editing

```
[1] 1 import numpy as np
2 np.random.seed(123)
3 import pandas as pd
4 import pandas.testing as tm
5 import pandas.util.testing as tm
6 import seaborn as sns
7 import warnings
8 warnings.filterwarnings("ignore")
9 from tqdm import tqdm_notebook
10 from statsmodels.stats.outliers_influence import variance_inflation_factor
11 from sklearn.model_selection import train_test_split
12 from sklearn.decomposition import TruncatedSVD
13 from sklearn.decomposition import PCA
14 from sklearn.decomposition import FastICA
15 from sklearn.ensemble import RandomForestRegressor
16 from sklearn.model_selection import RandomizedSearchCV
17 import matplotlib.pyplot as plt
18 from sklearn.metrics import r2_score
19 import xgboost as xgb
20 from sklearn.tree import DecisionTreeRegressor
21 from sklearn.linear_model import LinearRegression
22 import statsmodels.api as sm
23 from sklearn.model_selection import cross_val_score
24 from sklearn.model_selection import GridSearchCV
25 from sklearn.linear_model import SGDRegressor
26 from sklearn.linear_model import Ridge
27 from sklearn.linear_model import Lasso
28 from sklearn import linear_model
29 from sklearn import metrics
30 from sklearn.ensemble import AdaBoostRegressor
31 from sklearn.svm import SVR
32 import json
33 from tqdm import tqdm_notebook
34 from statsmodels.stats.outliers_influence import variance_inflation_factor
```

↳ /usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: FutureWarning: pandas.

```
[2] 1 # loading the train and test dataset
2 df_1=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Case Study 1/mercedes-benz-fault-diagnosis-maintenance.csv')
3 df_2=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Case Study 1/mercedes-benz-fault-diagnosis-maintenance.csv')
```

```
[3] 1 def final_pred(df_1, df_2):
2     # converting categorical features to numerical features
```

```

3     x_dummies_1=pd.get_dummies(df_1)
4     x_dummies_2=pd.get_dummies(df_2)
5
6     missing_cols = set( x_dummies_1.columns ) - set( x_dummies_2.columns )
7     y=x_dummies_1['y']
8     # align dataframes
9     x_dummies_1, x_dummies_2= x_dummies_1.align(x_dummies_2, join='inner', axis=1)
10
11    # creating a list of columns which have only zeros
12    zeros=[]
13    for i,j in x_dummies_1.any().items():#https://pandas.pydata.org/pandas-docs/stable/
14        if j==False:
15            zeros.append(i)
16
17    # dropping the columns containing only zeros
18    x_dummies_1 = x_dummies_1.drop(zeros, axis=1)
19    x_dummies_2 = x_dummies_2.drop(zeros, axis=1)
20    # saving the dataframes as a backup for further computations
21    x_1_safe = x_dummies_1
22    x_2_safe = x_dummies_2
23    x_dummies_1['y']=y
24    # Dropping Outliers
25    x_filtered= x_dummies_1[x_dummies_1['y']>70]https://www.geeksforgeeks.org/drop/
26    x_filtered= x_filtered[x_filtered['y']<150]
27    y=x_filtered['y']
28    x_filtered=x_filtered.drop(['y'], axis=1)
29    top_20_features= ['ID', 'X14', 'X29', 'X54', 'X76', 'X118', 'X119', 'X127', 'X136',
30                      'X189', 'X222', 'X232', 'X263', 'X279', 'X311', 'X314', 'X315',
31                      'X6_g', 'X6_j']
32    def calc_vif(X):
33
34        # Calculationg VIF
35        vif=pd.DataFrame()
36        vif[ 'variables']=X.columns
37        vif[ 'VIF']= [variance_inflation_factor(X.values, i) for i in tqdm_notebook
38
39        return (vif)
40    '''vif_values=calc_vif(x_filtered)
41    vif_values=vif_values.sort_values(by='VIF', ascending=True)
42    lst_variables= list(vif_values['variables'])
43    lst_vif= list(vif_values['VIF'])
44    vif_dict = dict(zip(lst_variables, lst_vif))'''
45    '''inf_indices=[]
46    count=0
47    for k,v in vif_dict.items():
48        if np.isinf(v):
49            inf_indices.append(k)
50            count+=1'''
51    '''# removing top_20_features from the features that need to be dropped from 'inf_indices'
52    inf_indices=list(set(inf_indices)-set(top_20_features))'''
53    with open('/content/drive/My Drive/Colab Notebooks/Case Study 1/inf_indices.json') as f:
54        checc=json.loads(f.read())
55    inf_indices=checc

```

```

56 # dropping the features
57 x_filtered = x_filtered.drop(inf_indices, axis=1)
58 x_dummies_2= x_dummies_2.drop(inf_indices, axis=1)
59 # generating SVD features
60 tsvd= TruncatedSVD(n_components=2, random_state=42)
61 tsvd_train= tsvd.fit_transform(x_filtered)
62 tsvd_test= tsvd.transform(x_dummies_2)
63 # generating PCA features
64 pca = PCA(n_components=2, random_state=42)
65 pca_train= pca.fit_transform(x_filtered)
66 pca_test= pca.transform(x_dummies_2)
67 # generating ICA features
68 ica=FastICA(n_components=2, random_state=42)
69 ica_train= ica.fit_transform(x_filtered)
70 ica_test= ica.transform(x_dummies_2)
71 # adding these dimensionality reduction features to the dataset
72 for i in range(0, tsvd_train.shape[1]):
73     x_filtered['tsvd_'+str(i)]= tsvd_train[:, i]
74     x_dummies_2['tsvd_'+str(i)]= tsvd_test[:, i]
75     x_filtered['pca_'+str(i)]= pca_train[:, i]
76     x_dummies_2['pca_'+str(i)]= pca_test[:, i]
77     x_filtered['ica_'+str(i)]= ica_train[:, i]
78     x_dummies_2['ica_'+str(i)]= ica_test[:, i]
79
80 # adding new features using feature interaction of important features
81 x_filtered['X64 + X218']=x_filtered['X64']+x_filtered['X218']
82 x_dummies_2['X64 + X218']=x_dummies_2['X64']+x_dummies_2['X218']
83
84 x_filtered['X218 + X224 + X273']=x_filtered['X218']+x_filtered['X224'] + x_fi
85 x_dummies_2['X218 + X224 + X273']=x_dummies_2['X218']+x_dummies_2['X224'] + x
86
87 x_filtered['X64 + X224 + X273']=x_filtered['X64']+x_filtered['X224'] + x_filt
88 x_dummies_2['X64 + X224 + X273']=x_dummies_2['X64']+x_dummies_2['X224'] + x_d
89
90 x_filtered['X314 + X315']=x_filtered['X314']+x_filtered['X315']
91 x_dummies_2['X314 + X315']=x_dummies_2['X314']+x_dummies_2['X315']
92
93 x_filtered['X314 + X315 + X29']=x_filtered['X314']+x_filtered['X315'] + x_fil
94 x_dummies_2['X314 + X315 + X29']=x_dummies_2['X314']+x_dummies_2['X315'] + x_d
95
96 x_filtered['X314 + X315 + X118']=x_filtered['X314']+x_filtered['X315'] + x_fi
97 x_dummies_2['X314 + X315 + X118']=x_dummies_2['X314']+x_dummies_2['X315'] + x_d
98
99 x_filtered['X127 + X189']=x_filtered['X127']+x_filtered['X189']
100 x_dummies_2['X127 + X189']=x_dummies_2['X127']+x_dummies_2['X189']
101
102 x_filtered['X118 + X54']=x_filtered['X118']+x_filtered['X54']
103 x_dummies_2['X118 + X54']=x_dummies_2['X118']+x_dummies_2['X54']
104
105 x_train, x_test, y_train, y_test = train_test_split(x_filtered, y, test_size=
106
107 # applying RandomForestRegressor model
108 # Number of trees in a Random forest

```

```

109     n_estimators=[int(x) for x in np.linspace (start=100, stop=1000, num=10)]
110
111     # Number of features to consider at every split
112     max_features=['auto', 'sqrt']
113
114     # Maximum no of levels in a tree
115     max_depth=[int(x) for x in np.linspace(10, 110, num = 11)]
116     max_depth.append(None)
117
118     # minimum number of samples required to split a node
119     min_samples_split = np.arange(50,250,20)
120
121     # Minimum number of samples required at each leaf node
122     min_samples_leaf = np.arange(5,50,5)
123     # Method of selecting samples for training each tree
124     bootstrap = [True, False]
125
126     # Create the random grid
127     random_grid = {'n_estimators': n_estimators,
128                     'max_features': max_features,
129                     'max_depth': max_depth,
130                     'min_samples_split': min_samples_split,
131                     'min_samples_leaf': min_samples_leaf,
132                     'bootstrap': bootstrap}
133
134     # First create the base model to tune
135     rf = RandomForestRegressor()
136
137     # Random search of parameters, using 3 fold cross validation,
138     # search across 100 different combinations, and use all available cores
139     '''rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
140     # Fit the random search model
141     rf_random.fit(x_filtered, y)
142     best_rf=rf_random.best_estimator_'''
143
144     best_rf= RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
145                                     max_depth=70, max_features='auto', max_leaf_nodes=None,
146                                     max_samples=None, min_impurity_decrease=0.0,
147                                     min_impurity_split=None, min_samples_leaf=40,
148                                     min_samples_split=110, min_weight_fraction_leaf=0.0,
149                                     n_estimators=500, n_jobs=None, oob_score=False,
150                                     random_state=None, verbose=0, warm_start=False)
151
152     # fitting the best model on to the dataset
153     best_rf.fit(x_filtered, y)
154
155     y_train_pred=best_rf.predict(x_filtered)
156     y_test_pred=best_rf.predict(x_dummies_2)
157
158     return y_train_pred, y_test_pred

```

[4] 1 y_train_pred, y_test_pred = final_pred(df_1, df_2)

[] 1

[5] 1 # loading the train and test dataset

```

2 df_1=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Case Study 1/mercedes-benz-fault-diagnosis-dataset.csv')
3 df_2=pd.read_csv('/content/drive/My Drive/Colab Notebooks/Case Study 1/mercedes-benz-fault-diagnosis-dataset.csv')
4 def final_metric(df_1, df_2):
5     # converting categorical features to numerical features
6     x_dummies_1=pd.get_dummies(df_1)
7     x_dummies_2=pd.get_dummies(df_2)
8
9     missing_cols = set( x_dummies_1.columns ) - set( x_dummies_2.columns )
10    y=x_dummies_1['y']
11    # align dataframes
12    x_dummies_1, x_dummies_2= x_dummies_1.align(x_dummies_2, join='inner', axis=1)
13
14    # creating a list of columns which have only zeros
15    zeros=[]
16    for i,j in x_dummies_1.any().items():#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.any.html
17        if j==False:
18            zeros.append(i)
19
20    # dropping the columns containing only zeros
21    x_dummies_1 = x_dummies_1.drop(zeros, axis=1)
22    x_dummies_2 = x_dummies_2.drop(zeros, axis=1)
23    # saving the dataframes as a backup for further computations
24    x_1_safe = x_dummies_1
25    x_2_safe = x_dummies_2
26    x_dummies_1['y']=y
27    # Dropping Outliers
28    x_filtered= x_dummies_1[x_dummies_1['y']>70]#https://www.geeksforgeeks.org/drop-outliers-in-pandas-dataframe/
29    x_filtered= x_filtered[x_filtered['y']<150]
30    y=x_filtered['y']
31    x_filtered=x_filtered.drop(['y'], axis=1)
32    top_20_features= ['ID', 'X14', 'X29', 'X54', 'X76', 'X118', 'X119', 'X127', 'X130', 'X136', 'X189', 'X222', 'X232', 'X263', 'X279', 'X311', 'X314', 'X315', 'X6_g', 'X6_j']
33
34    def calc_vif(X):
35
36        # Calculationg VIF
37        vif=pd.DataFrame()
38        vif['variables']=X.columns
39        vif[ 'VIF']= [variance_inflation_factor(X.values, i) for i in tqdm_notebook.tqdm(range(X.shape[1]))]
40
41        return (vif)
42
43    '''vif_values=calc_vif(x_filtered)
44    vif_values=vif_values.sort_values(by='VIF', ascending=True)
45    lst_variables= list(vif_values['variables'])
46    lst_vif= list(vif_values['VIF'])
47    vif_dict = dict(zip(lst_variables, lst_vif))'''
48    '''inf_indices=[]
49    count=0
50    for k,v in vif_dict.items():
51        if np.isinf(v):
52            inf_indices.append(k)
53            count+=1'''
54    '''# removing top_20_features from the features that need to be dropped from the dataframes
55    for i in top_20_features:
56        if i in inf_indices:
57            inf_indices.remove(i)
58
59    for i in inf_indices:
60        if i in lst_vif:
61            lst_vif.remove(i)
62
63    for i in lst_vif:
64        if i in lst_variables:
65            lst_variables.remove(i)
66
67    for i in lst_variables:
68        if i in vif_values['variables']:
69            vif_values=vif_values[vif_values['variables']!=i]
70
71    vif_values=vif_values.sort_values(by='VIF', ascending=True)
72    lst_variables= list(vif_values['variables'])
73    lst_vif= list(vif_values['VIF'])'''
```

```

55 inf_indices=list(set(inf_indices)-set(top_20_features))''
56 with open('/content/drive/My Drive/Colab Notebooks/Case Study 1/inf_indices.json') as f:
57     checc=json.loads(f.read())
58 inf_indices=checc
59 # dropping the features
60 x_filtered = x_filtered.drop(inf_indices, axis=1)
61 x_dummies_2= x_dummies_2.drop(inf_indices, axis=1)
62 # generating SVD features
63 tsvd= TruncatedSVD(n_components=2, random_state=42)
64 tsvd_train= tsvd.fit_transform(x_filtered)
65 tsvd_test= tsvd.transform(x_dummies_2)
66 # generating PCA features
67 pca = PCA(n_components=2, random_state=42)
68 pca_train= pca.fit_transform(x_filtered)
69 pca_test= pca.transform(x_dummies_2)
70 # generating ICA features
71 ica=FastICA(n_components=2, random_state=42)
72 ica_train= ica.fit_transform(x_filtered)
73 ica_test= ica.transform(x_dummies_2)
74 # adding these dimensionality reduction features to the dataset
75 for i in range(0, tsvd_train.shape[1]):
76     x_filtered['tsvd_'+str(i)]= tsvd_train[:, i]
77     x_dummies_2['tsvd_'+str(i)]= tsvd_test[:, i]
78     x_filtered['pca_'+str(i)]= pca_train[:, i]
79     x_dummies_2['pca_'+str(i)]= pca_test[:, i]
80     x_filtered['ica_'+str(i)]= ica_train[:, i]
81     x_dummies_2['ica_'+str(i)]= ica_test[:, i]
82
83 # adding new features using feature interaction of important features
84 x_filtered['X64 + X218']=x_filtered['X64']+x_filtered['X218']
85 x_dummies_2['X64 + X218']=x_dummies_2['X64']+x_dummies_2['X218']
86
87 x_filtered['X218 + X224 + X273']=x_filtered['X218']+x_filtered['X224'] + x_filtered['X273']
88 x_dummies_2['X218 + X224 + X273']=x_dummies_2['X218']+x_dummies_2['X224'] + x_dummies_2['X273']
89
90 x_filtered['X64 + X224 + X273']=x_filtered['X64']+x_filtered['X224'] + x_filtered['X273']
91 x_dummies_2['X64 + X224 + X273']=x_dummies_2['X64']+x_dummies_2['X224'] + x_dummies_2['X273']
92
93 x_filtered['X314 + X315']=x_filtered['X314']+x_filtered['X315']
94 x_dummies_2['X314 + X315']=x_dummies_2['X314']+x_dummies_2['X315']
95
96 x_filtered['X314 + X315 + X29']=x_filtered['X314']+x_filtered['X315'] + x_filtered['X29']
97 x_dummies_2['X314 + X315 + X29']=x_dummies_2['X314']+x_dummies_2['X315'] + x_dummies_2['X29']
98
99 x_filtered['X314 + X315 + X118']=x_filtered['X314']+x_filtered['X315'] + x_filtered['X118']
100 x_dummies_2['X314 + X315 + X118']=x_dummies_2['X314']+x_dummies_2['X315'] + x_dummies_2['X118']
101
102 x_filtered['X127 + X189']=x_filtered['X127']+x_filtered['X189']
103 x_dummies_2['X127 + X189']=x_dummies_2['X127']+x_dummies_2['X189']
104
105 x_filtered['X118 + X54']=x_filtered['X118']+x_filtered['X54']
106 x_dummies_2['X118 + X54']=x_dummies_2['X118']+x_dummies_2['X54']
107

```

```

108     x_train, x_test, y_train, y_test = train_test_split(x_filtered, y, test_size=1)
109
110     # applying RandomForestRegressor model
111     # Number of trees in a Random forest
112     n_estimators=[int(x) for x in np.linspace (start=100, stop=1000, num=10)]
113
114     # Number of features to consider at every split
115     max_features=['auto', 'sqrt']
116
117     # Maximum no of levels in a tree
118     max_depth=[int(x) for x in np.linspace(10, 110, num = 11)]
119     max_depth.append(None)
120
121     # minimum number of samples required to split a node
122     min_samples_split = np.arange(50,250,20)
123
124     # Minimum number of samples required at each leaf node
125     min_samples_leaf = np.arange(5,50,5)
126     # Method of selecting samples for training each tree
127     bootstrap = [True, False]
128
129     # Create the random grid
130     random_grid = {'n_estimators': n_estimators,
131                     'max_features': max_features,
132                     'max_depth': max_depth,
133                     'min_samples_split': min_samples_split,
134                     'min_samples_leaf': min_samples_leaf,
135                     'bootstrap': bootstrap}
136
137     # First create the base model to tune
138     rf = RandomForestRegressor()
139
140     # Random search of parameters, using 3 fold cross validation,
141     # search across 100 different combinations, and use all available cores
142     '''rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
143     n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1)
144
145     # Fit the random search model
146     rf_random.fit(x_filtered, y)
147
148     best_rf=rf_random.best_estimator_'''
149
150     best_rf= RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
151                                     max_depth=70, max_features='auto', max_leaf_nodes=None,
152                                     max_samples=None, min_impurity_decrease=0.0,
153                                     min_impurity_split=None, min_samples_leaf=40,
154                                     min_samples_split=110, min_weight_fraction_leaf=0.0,
155                                     n_estimators=500, n_jobs=None, oob_score=False,
156                                     random_state=None, verbose=0, warm_start=False)
157
158     # fitting the best model on to the dataset
159     best_rf.fit(x_filtered, y)
160
161     y_train_pred=best_rf.predict(x_filtered)
162     y_test_pred=best_rf.predict(x_dummies_2)
163
164
165     return r2_score(y, y_train_pred)
166
167

```

```
↳ 0.6535470835738704
```

```
[7] 1 final_df= pd.DataFrame(list(zip(df_2['ID'].values, y_test_pred)), columns=['ID',
```

```
[8] 1 final_df.head()
```

```
↳      ID          y
 0    1  78.505489
 1    2  94.308220
 2    3  78.433665
 3    4  78.505489
 4    5  113.414816
```



```
1 #final_df.to_csv(r'/content/drive/My Drive/Colab Notebooks/Case Study 1/RandomFor
```