```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.preprocessing import normalize, StandardScaler,
  PolynomialFeatures
5 from sklearn neural network import MLPRegressor
 6 from sklearn. model selection import cross val score
7 from sklearn. cluster import KMeans
8 from sklearn.ensemble import RandomForestRegressor,
  RandomForestClassifier
9 from sklearn. svm import SVR
10 from sklearn.metrics import mean absolute error
11
12 def feature engineering(x):
13 x ['matchDuration'] += 1
14 .... x["totalDistance"] = x["rideDistance"] + x["swimDistance"] + x[
   "walkDistance"] + 1
15 .... x["headshot_per_kill"] = x["headshotKills"] / x["kills"]
16 ... x["distance_per_second"] = x["totalDistance"] / x["
  matchDuration"]
17 .... x["boosts_per_distance"] = x["boosts"] / x["totalDistance"]
18 .... x["weapons per distance"] = x["weaponsAcquired"] / x["
  totalDistance"]
19 .... x["heals_per_distance"] = x["heals"] / x["totalDistance"]
20 ... x["heals_per_kill"] = x["heals"] / x["kills"]
21 ... x["killStreaks_per_second"] = x["killStreaks"] / x["
  matchDuration"
22 x ["boosts per kill"] = x ["boosts"] / x ["kills"]
23 x ["heals per second"] = x["heals"] / x["matchDuration"]
24 .... x["boosts_per_second"] = x["boosts"] / x["matchDuration"]
25 ... x["weapons_per_second"] = x["weaponsAcquired"] / x["
  matchDuration"]
26 ... x["walkDistance_percentile"] = x["walkDistance"] / x["
  totalDistance"
27 x["rideDistance percentile"] = x["rideDistance"] / x["
  totalDistance"
28 ... x["damage_per_second"] = x["damageDealt"] / x["matchDuration"]
29 x["damage_per_kill"] = x["damageDealt"] / x["kills"]
30 ... x["kill per distance"] = x["kills"] / x["totalDistance"]
31 ... x["weapons_per_kill"] = x["weaponsAcquired"] / x["kills"]
```

```
32 x. replace([np. inf, -np. inf], np. nan, inplace=True)
33 x. fillna(0, inplace=True)
34
35 x = pd. read csv("solo_test")
36 test = pd. read csv("solo test test")
37 test.drop(['Unnamed: 0', 'Id', 'groupId', 'matchId', 'killPoints','
  matchType', 'DBNOs', 'revives', 'vehicleDestroys',
'winPoints', 'rankPoints', 'assists', 'teamKills'], axis=1,
   inplace=True)
39 test.dropna(inplace=True)
40 x.drop(['Unnamed: 0', 'Id', 'groupId', 'matchId', 'killPoints', '
  matchType', 'DBNOs', 'revives', 'vehicleDestroys',
41 'winPoints', 'rankPoints', 'assists', 'teamKills', ], axis=1
  , inplace=True)
42 x. dropna(inplace=True)
43
44 feature engineering(x)
45 feature engineering(test)
46
47 X_dev_clf = x.loc[:, 'boosts': 'weapons_per_kill']
48 X dev clf. drop(['winPlacePerc'], axis=1, inplace=True)
49 y dev clf = x.loc[:, 'winPlacePerc']
50 X test clf = test.loc[:, 'boosts': 'weapons_per_kill']
51 X test clf. drop(['winPlacePerc'], axis=1, inplace=True)
52 y_test_clf = test.loc[:, 'winPlacePerc']
53
54 prereg1 = RandomForestRegressor(n estimators=15, criterion="mae",
  max depth=11, min samples leaf=0.0001,
            max_leaf_nodes=500, n_jobs=-1,
   random state=0, min samples split=0.0001,)
56 print("train prereg1.")
57 preregl. fit(X dev clf, y dev clf)
58 print("performance of preregl:", mean_absolute_error(y_test_clf,
  preregl.predict(X test clf)))
59 print("add new feature.")
60 C1 = prereg1. predict (X_dev_c1f)
61 X_dev_clf['C1'] = C1
62 C1 = preregl.predict(X test clf)
63 X test clf['C1'] = C1
64
```

```
65 prereg2 = RandomForestRegressor(n_estimators=15, criterion="mae",
  max_depth=11, min_samples_leaf=0.0001,
66 ..... max_leaf_nodes=500, n_jobs=-1,
  random state=0, min samples split=0.0001,)
67 print("train prereg2.")
68 prereg2. fit(X_dev_clf, y_dev_clf)
69 print("performance of prereg2:", mean_absolute_error(y_test_clf,
  prereg2.predict(X_test_clf)))
70 print("add new feature.")
71 C2 = prereg2. predict(X dev c1f)
72 X dev clf['C1'] = C2
73 C2 = prereg2.predict(X test clf)
74 X test clf['C1'] = C2
75
76 prereg3 = RandomForestRegressor(n estimators=15, criterion="mae",
  max_depth=11, min_samples_leaf=0.0001,
   max_leaf_nodes=500, n_jobs=-1,
77
  random state=0, min samples split=0.0001,)
78 print("train prereg3.")
79 prereg3. fit(X_dev_clf, y_dev_clf)
80 print("performance of prereg3:", mean_absolute_error(y_test_clf,
  prereg3.predict(X test clf)))
81 print("add new feature.")
82 C3 = prereg3. predict(X dev clf)
83 X dev clf['C1'] = C3
84 C3 = prereg3. predict(X test clf)
85 X test clf['C1'] = C3
86
87 prereg4 = RandomForestRegressor(n_estimators=15, criterion="mae",
  max_depth=11, min_samples_leaf=0.0001,
88 ..... max_leaf_nodes=500, n_jobs=-1,
  random state=0, min samples split=0.0001,)
89 print ("train prereg4.")
90 prereg4. fit(X dev clf, y dev clf)
91 print("performance of prereg4:", mean_absolute_error(y_test_clf,
  prereg4.predict(X test clf)))
92 print("add new feature.")
93 C4 = prereg4. predict(X dev clf)
94 X dev clf['C1'] = C4
95 C4 = prereg4. predict(X test clf)
```

```
96 X test clf['C1'] = C4
97
98 prereg5 = RandomForestRegressor(n_estimators=15, criterion="mae",
   max depth=11, min samples leaf=0.0001,
99
             max_leaf_nodes=500, n_jobs=-1,
   random state=0, min samples split=0.0001,)
100 print ("train prereg5.")
101 prereg5. fit (X_dev_clf, y_dev_clf)
102 print ("performance of prereg5:", mean_absolute_error(y_test_clf,
   prereg5.predict(X test clf)))
103 print ("add new feature.")
104 C5 = prereg5. predict(X dev clf)
105 X dev clf['C1'] = C5
106 C5 = prereg5. predict(X test clf)
107 X test clf['C1'] = C5
108
109 prereg6 = RandomForestRegressor(n_estimators=15, criterion="mae",
   max depth=11, min samples leaf=0.0001,
          max_leaf_nodes=500, n jobs=-1,
110
   random_state=0, min_samples_split=0.0001,)
111 print("train prereg6.")
112 prereg6. fit (X dev clf, y dev clf)
113 print ("performance of prereg6:", mean absolute error (y test clf,
   prereg6.predict(X test clf)))
114 print ("add new feature.")
115 C6 = prereg6. predict(X dev clf)
116 X dev clf['C1'] = C6
117 C6 = prereg6. predict(X_test_clf)
118 X test clf['C1'] = C6
119
120 prereg7 = RandomForestRegressor(n estimators=15, criterion="mae",
   max depth=11, min samples leaf=0.0001,
121 ..... max_leaf_nodes=500, n_jobs=-1,
   random state=0, min samples split=0.0001,)
122 print("train prereg7.")
123 prereg7. fit (X_dev_clf, y_dev_clf)
124 print ("performance of prereg7:", mean absolute error (y test clf,
   prereg6.predict(X test clf)))
125 print ("add new feature.")
126 C7 = prereg7. predict(X dev clf)
```

```
127 X dev clf['C1'] = C7
128 C7 = prereg7. predict(X test clf)
129 X_test clf['C1'] = C7
130
131 prereg8 = RandomForestRegressor(n estimators=15, criterion="mae",
   max depth=11, min samples leaf=0.0001,
132 ..... max_leaf_nodes=500, n_jobs=-1,
   random_state=0, min_samples_split=0.0001,)
133 print("train prereg8.")
134 prereg8. fit (X dev clf, y dev clf)
135 print ("performance of prereg8:", mean absolute error (y test clf,
   prereg6.predict(X_test clf)))
136 print ("add new feature.")
137 C8 = prereg8. predict(X dev clf)
138 X dev clf['C1'] = C8
139 C8 = prereg8. predict(X_test_clf)
140 X test clf['C1'] = C8
141
142 prereg9 = RandomForestRegressor(n estimators=15, criterion="mae",
   max_depth=11, min_samples_leaf=0.0001,
143 ..... max_leaf_nodes=500, n_jobs=-1,
   random state=0, min samples split=0.0001,)
144 print ("train prereg9.")
145 prereg9. fit (X dev clf, y dev clf)
146 print ("performance of prereg9:", mean_absolute_error(y_test_clf,
   prereg6.predict(X test clf)))
147 print ("add new feature.")
148 C9 = prereg9. predict(X_dev_clf)
149 X dev clf['C1'] = C9
150 C9 = prereg9. predict(X_test_clf)
151 X test clf['C1'] = C9
152
153 prereg10 = RandomForestRegressor(n estimators=15, criterion="mae",
   max depth=11, min samples leaf=0.0001,
random state=0, min samples split=0.0001,)
155 print ("train prereg10.")
156 prereg10. fit(X dev clf, y dev clf)
157 print("performance of prereg10:", mean_absolute_error(y_test_clf,
   prereg6.predict(X test clf)))
```

```
158 print("add new feature.")
159 C10 = prereg10. predict(X dev c1f)
160 X_dev_clf['C1'] = C10
161 C10 = prereg10.predict(X test c1f)
162 X test clf['C1'] = C10
163
164 preregl1 = RandomForestRegressor(n_estimators=15, criterion="mae",
   max_depth=11, min_samples_leaf=0.0001,
165 ..... max_leaf_nodes=500, n_jobs=-1,
   random state=0, min samples split=0.0001,)
166 print ("train preregl1.")
167 preregll. fit (X dev clf, y dev clf)
168 print ("performance of preregl1:", mean_absolute_error(y_test_clf,
   prereg6.predict(X test clf)))
169 print ("add new feature.")
170 C11 = prereg11.predict(X_dev_c1f)
171 X dev clf['C1'] = C11
172 Cll = preregll.predict(X test clf)
173 X test clf['C1'] = C11
174
175 print("train reg.")
176 reg = RandomForestRegressor(n estimators=15, criterion="mae",
   max depth=15, min samples leaf=0.0001,
177 ..... max_leaf_nodes=500, n_jobs=-1,
   random_state=0, min_samples_split=0.0001,)
178 reg. fit (X dev clf, y dev clf)
179 scores = cross val score(reg, X dev clf, y dev clf, scoring="
   neg_mean_absolute_error", cv=2)
180 print(scores.mean(), scores.std())
181 print ("final performance:", mean absolute error (y test clf, reg.
   predict(X test clf)))
182
183 #X test clf['C'] = (X \text{ test } clf['Cl']/0.1). \text{ astype}(int)
184 #X dev = x.loc[:, 'assists': 'weaponsAcquired']
185 #y_dev = x.loc[:, 'winPlacePerc']
186 '''
187 \; model = []
188 for i in range(1, 12):
        X_{dev_{seg}} = X_{dev_{clf.loc}[(x['winPlacePerc']>=(0.1*(i-1)))&(
189
   x['winPlacePerc'] < (0.1*(i)))
```

```
y_{dev_seg} = y_{dev_clf.loc[(x['winPlacePerc']>=(0.1*(i-1)))&(
190
   x['winPlacePerc'] < (0.1*(i)))
191
       print("training the", i, "th regressor")
192
       reg = RandomForestRegressor(n_estimators=15, criterion="mae
193
    ", max_depth=11, min_samples_leaf=0.0001,
194
                                     max_leaf_nodes=300, n_jobs=-1,
   random_state=0, min_samples_split=0.0001,)
       reg. fit(X_dev_seg, y_dev_seg)
195
196
        model.append(reg)
197
        #scores = cross_val_score(reg, X_dev_seg, y_dev_seg, scoring
   ="neg_mean_absolute_error", cv=3)
       #print("score:", -scores.mean(), "(mean)", scores.std(), "(
198
   std)")
199 y_pre = []
200 X_test_clf = X_test_clf.values
201 for index, record in enumerate(X_test_clf):
202
        i = record[len(record)-1]/0.1
203
       print(index)
204
        y_{pre.append (model[int(i)].predict(record.reshape(1, -1))[0]
   ])
205 print ("final performance:", mean absolute error (y test clf,
206 ***
```