```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize, StandardScaler, PolynomialFeatures
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_score
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error

def feature_engineering(x):
    x['matchDuration'] += 1
    x["totalDistance"] = x["rideDistance"] + x["swimDistance"] + x["walkDistance"] + 1
    x["headshot_per_kill"] = x["headshotKills"] / x["kills"]
    x["distance_per_second"] = x["totalDistance"] / x["matchDuration"]
    x["boosts_per_distance"] = x["boosts"] / x["totalDistance"]
    x["weapons_per_distance"] = x["weaponsAcquired"] / x["totalDistance"]
    x["heals_per_distance"] = x["heals"] / x["totalDistance"]
    x["heals_per_kill"] = x["heals"] / x["kills"]
    x["killStreaks_per_second"] = x["killStreaks"] / x["matchDuration"]
    x["boosts_per_kill"] = x["boosts"] / x["kills"]
    x["heals_per_second"] = x["heals"] / x["matchDuration"]
    x["boosts_per_second"] = x["boosts"] / x["matchDuration"]
    x["weapons_per_second"] = x["weaponsAcquired"] / x["matchDuration"]
    x["walkDistance_percentile"] = x["walkDistance"] / x["totalDistance"]
    x["rideDistance_percentile"] = x["rideDistance"] / x["totalDistance"]
    x["damage_per_second"] = x["damageDealt"] / x["matchDuration"]
    x["damage_per_kill"] = x["damageDealt"] / x["kills"]
    x["kill_per_distance"] = x["kills"] / x["totalDistance"]
    x["weapons_per_kill"] = x["weaponsAcquired"] / x["kills"]
    x.replace([np.inf, -np.inf], np.nan, inplace=True)
    x.fillna(0, inplace=True)

x = pd.read_csv("solo_test")
test = pd.read_csv("solo_test_test")
test.drop(['Unnamed: 0', 'Id','groupId','matchId','killPoints','matchType', 'DBNOs', 'revives', 'vehicleDestroys',
        'winPoints','rankPoints', 'assists', 'teamKills'], axis=1, inplace=True)
test.dropna(inplace=True)
x.drop(['Unnamed: 0', 'Id','groupId','matchId','killPoints','matchType', 'DBNOs', 'revives', 'vehicleDestroys',
        'winPoints','rankPoints', 'assists', 'teamKills', ], axis=1, inplace=True)
x.dropna(inplace=True)

feature_engineering(x)
feature_engineering(test)

X_dev_clf = x.loc[:, 'boosts':'weapons_per_kill']
X_dev_clf.drop(['winPlacePerc'], axis=1, inplace=True)
y_dev_clf = x.loc[:, 'winPlacePerc']
X_test_clf = test.loc[:, 'boosts':'weapons_per_kill']
X_test_clf.drop(['winPlacePerc'], axis=1, inplace=True)
y_test_clf = test.loc[:, 'winPlacePerc']

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,)
print("train prereg1.")
prereg1.fit(X_dev_clf, y_dev_clf)
print("performance of prereg1:", mean_absolute_error(y_test_clf, prereg1.predict(X_test_clf)))
print("add new feature.")
C1 = prereg1.predict(X_dev_clf)
X_dev_clf['C1'] = C1
C1 = prereg1.predict(X_test_clf)
X_test_clf['C1'] = C1

```

```python
prereg2 = 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,)
print("train prereg2.")
prereg2.fit(X_dev_clf, y_dev_clf)
print("performance of prereg2:", mean_absolute_error(y_test_clf, prereg2.predict(X_test_clf)))
print("add new feature.")
C2 = prereg2.predict(X_dev_clf)
X_dev_clf['C1'] = C2
C2 = prereg2.predict(X_test_clf)
X_test_clf['C1'] = C2

prereg3 = 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,)
print("train prereg3.")
prereg3.fit(X_dev_clf, y_dev_clf)
print("performance of prereg3:", mean_absolute_error(y_test_clf, prereg3.predict(X_test_clf)))
print("add new feature.")
C3 = prereg3.predict(X_dev_clf)
X_dev_clf['C1'] = C3
C3 = prereg3.predict(X_test_clf)
X_test_clf['C1'] = C3

prereg4 = 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,)
print("train prereg4.")
prereg4.fit(X_dev_clf, y_dev_clf)
print("performance of prereg4:", mean_absolute_error(y_test_clf, prereg4.predict(X_test_clf)))
print("add new feature.")
C4 = prereg4.predict(X_dev_clf)
X_dev_clf['C1'] = C4
C4 = prereg4.predict(X_test_clf)
```

```
X_test_clf['C1'] = C4

prereg5 = 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,)
print("train prereg5.")
prereg5.fit(X_dev_clf, y_dev_clf)
print("performance of prereg5:", mean_absolute_error(y_test_clf, prereg5.predict(X_test_clf)))
print("add new feature.")
C5 = prereg5.predict(X_dev_clf)
X_dev_clf['C1'] = C5
C5 = prereg5.predict(X_test_clf)
X_test_clf['C1'] = C5

prereg6 = 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,)
print("train prereg6.")
prereg6.fit(X_dev_clf, y_dev_clf)
print("performance of prereg6:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C6 = prereg6.predict(X_dev_clf)
X_dev_clf['C1'] = C6
C6 = prereg6.predict(X_test_clf)
X_test_clf['C1'] = C6

prereg7 = 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,)
print("train prereg7.")
prereg7.fit(X_dev_clf, y_dev_clf)
print("performance of prereg7:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C7 = prereg7.predict(X_dev_clf)
```

```python
X_dev_clf['C1'] = C7
C7 = prereg7.predict(X_test_clf)
X_test_clf['C1'] = C7

prereg8 = 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,)
print("train prereg8.")
prereg8.fit(X_dev_clf, y_dev_clf)
print("performance of prereg8:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C8 = prereg8.predict(X_dev_clf)
X_dev_clf['C1'] = C8
C8 = prereg8.predict(X_test_clf)
X_test_clf['C1'] = C8

prereg9 = 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,)
print("train prereg9.")
prereg9.fit(X_dev_clf, y_dev_clf)
print("performance of prereg9:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C9 = prereg9.predict(X_dev_clf)
X_dev_clf['C1'] = C9
C9 = prereg9.predict(X_test_clf)
X_test_clf['C1'] = C9

prereg10 = 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,)
print("train prereg10.")
prereg10.fit(X_dev_clf, y_dev_clf)
print("performance of prereg10:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C10 = prereg10.predict(X_dev_clf)
X_dev_clf['C1'] = C10
C10 = prereg10.predict(X_test_clf)
X_test_clf['C1'] = C10

prereg11 = 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,)
print("train prereg11.")
prereg11.fit(X_dev_clf, y_dev_clf)
print("performance of prereg11:", mean_absolute_error(y_test_clf, prereg6.predict(X_test_clf)))
print("add new feature.")
C11 = prereg11.predict(X_dev_clf)
X_dev_clf['C1'] = C11
C11 = prereg11.predict(X_test_clf)
X_test_clf['C1'] = C11

print("train reg.")
reg = RandomForestRegressor(n_estimators=15, criterion="mae", max_depth=15, min_samples_leaf=0.0001,
                                max_leaf_nodes=500, n_jobs=-1, random_state=0, min_samples_split=0.0001,)
reg.fit(X_dev_clf, y_dev_clf)
scores = cross_val_score(reg, X_dev_clf, y_dev_clf, scoring="neg_mean_absolute_error", cv=2)
print(scores.mean(), scores.std())
print("final performance:", mean_absolute_error(y_test_clf, reg.predict(X_test_clf)))

#X_test_clf['C'] = (X_test_clf['C1']/0.1).astype(int)
#X_dev = x.loc[:, 'assists': 'weaponsAcquired']
#y_dev = x.loc[:, 'winPlacePerc']
'''
model = []
for i in range(1, 12):
    X_dev_seg = X_dev_clf.loc[(x['winPlacePerc']>=(0.1*(i-1)))&(x['winPlacePerc']<(0.1*(i)))]
```

```python
    y_dev_seg = y_dev_clf.loc[(x['winPlacePerc']>=(0.1*(i-1)))&(x['winPlacePerc']<(0.1*(i)))]

    print("training the", i, "th regressor")
    reg = RandomForestRegressor(n_estimators=15, criterion="mae", max_depth=11, min_samples_leaf=0.0001,
                                max_leaf_nodes=300, n_jobs=-1, random_state=0, min_samples_split=0.0001,)
    reg.fit(X_dev_seg, y_dev_seg)
    model.append(reg)
    #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(), "(std)")
y_pre = []
X_test_clf = X_test_clf.values
for index, record in enumerate(X_test_clf):
    i = record[len(record)-1]/0.1
    print(index)
    y_pre.append(model[int(i)].predict(record.reshape(1, -1))[0])
print("final performance:", mean_absolute_error(y_test_clf, y_pre))
'''
```