```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize, StandardScaler, 
PolynomialFeatures
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import RandomForestRegressor

data = pd.read_csv("solo_sample")
data.drop(['Unnamed: 0', 'Id','groupId','matchId','killPoints','
matchType', 'DBNOs', 'revives', 'vehicleDestroys',
        'winPoints','rankPoints'], axis=1, inplace=True)
data.dropna(inplace=True)

shuffled_indices = np.random.permutation(len(data))
test_set_size = int(len(data) * 0.3)
test_indices = shuffled_indices[:test_set_size]
train_indices = shuffled_indices[test_set_size:]
dev_set=data.iloc[train_indices]
test_set=data.iloc[test_indices]

X_dev = dev_set.loc[:, 'assists' : 'weaponsAcquired']
y_dev = dev_set.loc[:, 'winPlacePerc']
X_test = test_set.loc[:, 'assists' : 'weaponsAcquired']
y_test =test_set.loc[:, 'winPlacePerc']

X_dev_normalized = normalize(X_dev, norm='l2', axis=0)
X_test_normalized = normalize(X_test, norm='l2', axis=0)
scaler = StandardScaler().fit(X_dev_normalized)
X_dev_final = scaler.transform(X_dev_normalized)
X_test_final = scaler.transform(X_test_normalized)

poly_features = PolynomialFeatures(degree=2, include_bias=True)
dev = poly_features.fit_transform(X_dev_final)
test = poly_features.fit_transform(X_test_final)

X_dev_normalized = normalize(dev, norm='l2', axis=0)
X_test_normalized = normalize(test, norm='l2', axis=0)
scaler = StandardScaler().fit(X_dev_normalized)
```

```
X_dev_final = scaler.transform(X_dev_normalized)
X_test_final = scaler.transform(X_test_normalized)

reg1 = MLPRegressor(hidden_layer_sizes=(10,10,10), max_iter=10000, solver='adam', verbose=True, tol=1e-8, random_state=1,
                    learning_rate_init=0.0015, n_iter_no_change=20, batch_size=int(len(X_dev_final)/75), alpha=1e-3)
reg2 = RandomForestRegressor(n_estimators=15, criterion="mae", max_depth=10, min_samples_leaf=0.001,
                             max_leaf_nodes=300, n_jobs=-1, random_state=0, min_samples_split=0.001,)
reg1.fit(X_dev_final, y_dev)
reg2.fit(X_dev_final, y_dev)
#scores = cross_val_score(NN, X_dev_standardized, y_dev, scoring="neg_mean_absolute_error", cv=2)
y_pre1 = reg1.predict(X_test_final)
score1 = mean_absolute_error(y_test, y_pre1)
print("mlp:", score1)
y_pre2 = reg2.predict(X_test_final)
score2 = mean_absolute_error(y_test, y_pre2)
print("rfr:", score2)
y_pre = (y_pre1+y_pre2)/2
score = mean_absolute_error(y_test, y_pre)
print("stack:", score)
```