```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

from sklearn import datasets
wine = datasets.load_wine()
wine_df = pd.DataFrame(wine.data, columns=wine.feature_names)
wine_df.head()
```


```
{"summary":"{\n  \"name\": \"wine_df\",\n  \"rows\": 178,\n
\"fields\": [\n    {\n      \"column\": \"alcohol\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.8118265380058575,\n        \"min\": 11.03,\n        \"max\": 14.83,\
n        \"num_unique_values\": 126,\n        \"samples\": [\n
11.62,\n          13.64,\n          13.69\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"malic_acid\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.1171460976144627,\n        \"min\": 0.74,\n        \"max\": 5.8,\n
\"num_unique_values\": 133,\n        \"samples\": [\n          1.21,\n
2.83,\n          1.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ash\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.27434400906081485,\n        \"min\": 1.36,\n
\"max\": 3.23,\n        \"num_unique_values\": 79,\n
\"samples\": [\n          2.31,\n          2.43,\n          2.52\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"alcalinity_of_ash\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3.339563767173505,\n        \"min\": 10.6,\n        \"max\": 30.0,\n
\"num_unique_values\": 63,\n        \"samples\": [\n          25.5,\n
28.5,\n          15.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"magnesium\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.282483515295665,\n        \"min\":
70.0,\n        \"max\": 162.0,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          126.0,\n          85.0,\n          162.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"total_phenols\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.6258510488339893,\n        \"min\": 0.98,\n        \"max\": 3.88,\n
\"num_unique_values\": 97,\n        \"samples\": [\n          1.68,\n
2.11,\n          1.35\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flavanoids\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.9988586850169467,\n        \"min\":
0.34,\n        \"max\": 5.08,\n        \"num_unique_values\": 132,\n
```

\"samples\": [\n   3.18,\n   2.5,\n   3.17\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"nonflavanoid_phenols\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.12445334029667937,\n   \"min\": 0.13,\n   \"max\": 0.66,\n   \"num_unique_values\": 39,\n   \"samples\": [\n   0.58,\n   0.41,\n   0.39\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"proanthocyanins\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.5723588626747613,\n   \"min\": 0.41,\n   \"max\": 3.58,\n   \"num_unique_values\": 101,\n   \"samples\": [\n   0.75,\n   1.77,\n   1.42\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"color_intensity\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 2.318285871822413,\n   \"min\": 1.28,\n   \"max\": 13.0,\n   \"num_unique_values\": 132,\n   \"samples\": [\n   2.95,\n   3.3,\n   5.1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"hue\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.22857156582982338,\n   \"min\": 0.48,\n   \"max\": 1.71,\n   \"num_unique_values\": 78,\n   \"samples\": [\n   1.22,\n   1.04,\n   1.45\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"od280/od315_of_diluted_wines\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.7099904287650504,\n   \"min\": 1.27,\n   \"max\": 4.0,\n   \"num_unique_values\": 122,\n   \"samples\": [\n   4.0,\n   1.82,\n   1.59\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"proline\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 314.9074742768491,\n   \"min\": 278.0,\n   \"max\": 1680.0,\n   \"num_unique_values\": 121,\n   \"samples\": [\n   1375.0,\n   1270.0,\n   735.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"wine_df"}

```
wine_df.isna().sum()
```

```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
```

```
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
```

```
# Разделение на объекты-признаки и целевой признак
X = wine_df.iloc[:, :-1].values
y = wine_df.iloc[:, -1].values

# Формирование обучающей и тестовой выборки
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 1)

from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error

# Создание экземпляра класса KNeighborsRegressor с K=15
knn = KNeighborsRegressor(n_neighbors=15)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# MAE - средняя абсолютная ошибка
mae = mean_absolute_error(y_test, y_pred)
# RMSE - среднеквадратичная ошибка (более чувствителен к наблюдением
далеким от среднего)
rmse = mean_squared_error(y_test, y_pred, squared=False)
# MAPE - средняя абсолютная ошибка в процентах
mape = mean_absolute_percentage_error(y_test, y_pred)

print("MAE:", mae)
print("RMSE:", rmse)
print("MAPE:", mape)
```

```
MAE: 231.86111111111111
RMSE: 281.1189043715519
MAPE: 0.3737341418083258
```

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV,
KFold, ShuffleSplit
from sklearn.metrics import mean_squared_error

knn = KNeighborsRegressor()

param_grid = {'n_neighbors': np.arange(1, 31)}

cv_strategies = [KFold(n_splits=5, shuffle=True, random_state=42),
                 ShuffleSplit(n_splits=5, test_size=0.2,
random_state=42)]
```

```python
# GridSearchCV
grid_search = GridSearchCV(knn, param_grid,
scoring='neg_mean_absolute_error', cv=cv_strategies[0])
grid_search.fit(X_train, y_train)

print("GridSearchCV Best MAE:", -grid_search.best_score_)
print("GridSearchCV Best Params:", grid_search.best_params_)

# RandomizedSearchCV
random_search = RandomizedSearchCV(knn, param_grid,
scoring='neg_mean_absolute_error', cv=cv_strategies[1], n_iter=10,
random_state=42)
random_search.fit(X_train, y_train)

print("RandomizedSearchCV Best MAE:", -random_search.best_score_)
print("RandomizedSearchCV Best Params:", random_search.best_params_)

# Обучение модели с лучшими параметрами GridSearchCV
best_model_gs = grid_search.best_estimator_
best_model_gs.fit(X_train, y_train)

# Предсказания на тестовом наборе данных
y_pred_gs = best_model_gs.predict(X_test)

# Вычисление RMSE
rmse_gs = np.sqrt(mean_squared_error(y_test, y_pred_gs))

# Вычисление MAPE
mape_gs = mean_absolute_percentage_error(y_test, y_pred_gs)

print("GridSearchCV Best RMSE:", rmse_gs)
print("GridSearchCV Best MAPE:", mape_gs)

# Обучение модели с лучшими параметрами RandomizedSearchCV
best_model_rs = random_search.best_estimator_
best_model_rs.fit(X_train, y_train)

# Предсказания на тестовом наборе данных
y_pred_rs = best_model_rs.predict(X_test)

# Вычисление RMSE
rmse_rs = np.sqrt(mean_squared_error(y_test, y_pred_rs))

# Вычисление MAPE
mape_rs = mean_absolute_percentage_error(y_test, y_pred_rs)

print("RandomizedSearchCV Best RMSE:", rmse_rs)
print("RandomizedSearchCV Best MAPE:", mape_rs)
```

```
GridSearchCV Best MAE: 171.22814039408868
GridSearchCV Best Params: {'n_neighbors': 8}
```

```
RandomizedSearchCV Best MAE: 171.7501915708812
RandomizedSearchCV Best Params: {'n_neighbors': 9}
GridSearchCV Best RMSE: 262.12011284719796
GridSearchCV Best MAPE: 0.3556902543653036
RandomizedSearchCV Best RMSE: 252.31200542098873
RandomizedSearchCV Best MAPE: 0.3453719388489157
```