# Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей

Цель лабораторной работы: изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

## Задание

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
4. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации. Сравните метрики качества исходной и оптимальной моделей.

## Ход работы

### Выбор и загрузка датасета

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.datasets import \*  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score  
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier   
from sklearn.model\_selection import train\_test\_split, GridSearchCV, RandomizedSearchCV  
from sklearn.model\_selection import cross\_val\_score, cross\_validate  
from sklearn.model\_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold  
  
def make\_dataframe(ds\_function):  
 ds = ds\_function()  
 df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],  
 columns= list(ds['feature\_names']) + ['target'])  
 return df  
  
wine = load\_wine()  
  
df = make\_dataframe(load\_wine)

# Первые 5 строк датасета  
df.head()

alcohol malic\_acid ash alcalinity\_of\_ash magnesium total\_phenols \  
0 14.23 1.71 2.43 15.6 127.0 2.80   
1 13.20 1.78 2.14 11.2 100.0 2.65   
2 13.16 2.36 2.67 18.6 101.0 2.80   
3 14.37 1.95 2.50 16.8 113.0 3.85   
4 13.24 2.59 2.87 21.0 118.0 2.80   
  
 flavanoids nonflavanoid\_phenols proanthocyanins color\_intensity hue \  
0 3.06 0.28 2.29 5.64 1.04   
1 2.76 0.26 1.28 4.38 1.05   
2 3.24 0.30 2.81 5.68 1.03   
3 3.49 0.24 2.18 7.80 0.86   
4 2.69 0.39 1.82 4.32 1.04   
  
 od280/od315\_of\_diluted\_wines proline target   
0 3.92 1065.0 0.0   
1 3.40 1050.0 0.0   
2 3.17 1185.0 0.0   
3 3.45 1480.0 0.0   
4 2.93 735.0 0.0

# Проверим наличие пустых значений  
# Цикл по колонкам датасета  
for col in df.columns:  
 # Количество пустых значений - все значения заполнены  
 temp\_null\_count = df[df[col].isnull()].shape[0]  
 print('{} - {}'.format(col, temp\_null\_count))

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  
target - 0

# Масштабирование  
scaler = MinMaxScaler()  
df[['alcohol', 'hue']] = scaler.fit\_transform(df[['alcohol', 'hue']])  
data = df[['alcohol','hue','target']]  
# data = pd.DataFrame(scaler.transform(df[['alcohol', 'hue']]), columns = ['alcohol', 'hue'])

data.head()

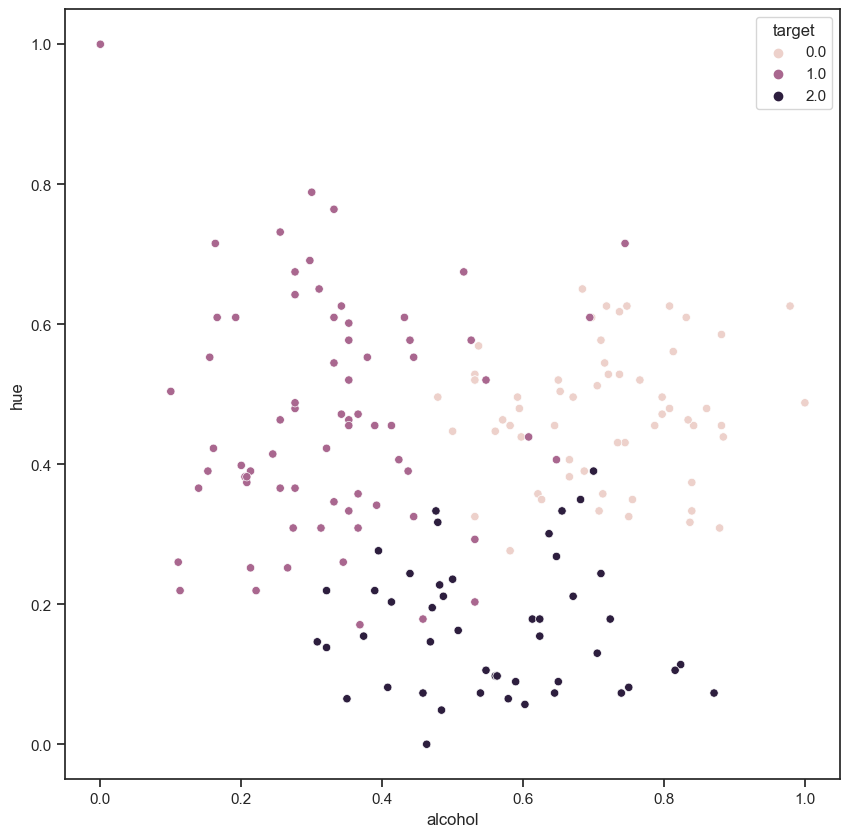
alcohol hue target  
0 0.842105 0.455285 0.0  
1 0.571053 0.463415 0.0  
2 0.560526 0.447154 0.0  
3 0.878947 0.308943 0.0  
4 0.581579 0.455285 0.0

data.describe().T

count mean std min 25% 50% 75% max  
alcohol 178.0 0.518584 0.213639 0.0 0.350658 0.531579 0.696711 1.0  
hue 178.0 0.388170 0.185831 0.0 0.245935 0.394309 0.520325 1.0  
target 178.0 0.938202 0.775035 0.0 0.000000 1.000000 2.000000 2.0

# Диаграмма рассеяния позволяет визуально обнаружить наличие зависимости  
# Построим зависимость между крепкостью алкоголя и оттенком цвета  
fig, ax = plt.subplots(figsize=(10,10))  
sns.scatterplot(ax=ax, x='alcohol', y='hue', data=data, hue='target')

<Axes: xlabel='alcohol', ylabel='hue'>



### Разделение на тестовую и обучающую

y = data['target']  
x = data.drop('target', axis = 1)  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.20, random\_state = 20)  
  
print(f"Обучающая выборка:\n{x\_train, y\_train}")  
print(f"Тестовая выборка:\n{x\_test, y\_test}")

Обучающая выборка:  
( alcohol hue  
1 0.571053 0.463415  
67 0.352632 0.520325  
174 0.623684 0.178862  
64 0.300000 0.788618  
20 0.797368 0.495935  
.. ... ...  
148 0.602632 0.056911  
137 0.394737 0.276423  
156 0.739474 0.073171  
15 0.684211 0.650407  
99 0.331579 0.764228  
  
[142 rows x 2 columns], 1 0.0  
67 1.0  
174 2.0  
64 1.0  
20 0.0  
 ...   
148 2.0  
137 2.0  
156 2.0  
15 0.0  
99 1.0  
Name: target, Length: 142, dtype: float64)  
Тестовая выборка:  
( alcohol hue  
111 0.392105 0.341463  
177 0.815789 0.105691  
46 0.881579 0.455285  
112 0.192105 0.609756  
91 0.255263 0.463415  
72 0.647368 0.406504  
140 0.500000 0.235772  
107 0.444737 0.325203  
125 0.273684 0.308943  
169 0.623684 0.154472  
47 0.755263 0.349593  
165 0.710526 0.243902  
120 0.110526 0.260163  
62 0.694737 0.609756  
13 0.978947 0.626016  
92 0.436842 0.390244  
87 0.163158 0.715447  
28 0.747368 0.626016  
51 0.736842 0.617886  
155 0.563158 0.097561  
161 0.700000 0.390244  
44 0.531579 0.325203  
124 0.221053 0.219512  
34 0.652632 0.504065  
132 0.468421 0.146341  
77 0.213158 0.390244  
45 0.836842 0.317073  
136 0.321053 0.219512  
80 0.255263 0.731707  
103 0.207895 0.373984  
68 0.607895 0.439024  
109 0.152632 0.390244  
152 0.547368 0.105691  
100 0.276316 0.642276  
12 0.715789 0.544715  
55 0.665789 0.406504, 111 1.0  
177 2.0  
46 0.0  
112 1.0  
91 1.0  
72 1.0  
140 2.0  
107 1.0  
125 1.0  
169 2.0  
47 0.0  
165 2.0  
120 1.0  
62 1.0  
13 0.0  
92 1.0  
87 1.0  
28 0.0  
51 0.0  
155 2.0  
161 2.0  
44 0.0  
124 1.0  
34 0.0  
132 2.0  
77 1.0  
45 0.0  
136 2.0  
80 1.0  
103 1.0  
68 1.0  
109 1.0  
152 2.0  
100 1.0  
12 0.0  
55 0.0  
Name: target, dtype: float64)

### Обучение и оценка качества модели для произвольного гиперпараметра K

cl1\_5 = KNeighborsClassifier(n\_neighbors=5).fit(x\_train, y\_train)  
  
target1\_5\_train = cl1\_5.predict(x\_train)  
target1\_5\_test = cl1\_5.predict(x\_test)  
  
accuracy\_score(y\_train, target1\_5\_train), accuracy\_score(y\_test, target1\_5\_test)

(0.9436619718309859, 0.8333333333333334)

Чем выше значения - тем лучше

### Подбор гиперпараметров модели и кросс-валидация

#### Grid Search

n\_range = np.array(range(5,31,1))  
tuned\_parameters = [{'n\_neighbors': n\_range}]  
tuned\_parameters

[{'n\_neighbors': array([ 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,  
 22, 23, 24, 25, 26, 27, 28, 29, 30])}]

%%time  
clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')  
clf\_gs.fit(x\_train, y\_train)

CPU times: user 347 ms, sys: 6.8 ms, total: 354 ms  
Wall time: 346 ms

GridSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_grid=[{'n\_neighbors': array([ 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,  
 22, 23, 24, 25, 26, 27, 28, 29, 30])}],  
 scoring='accuracy')

# Лучшая модель  
clf\_gs.best\_estimator\_

KNeighborsClassifier(n\_neighbors=13)

# Лучшее значение метрики  
clf\_gs.best\_score\_

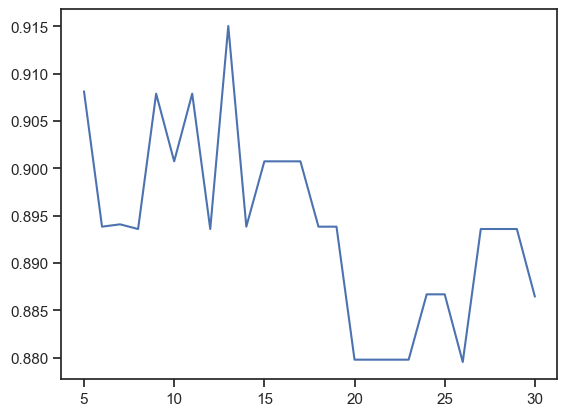
0.9150246305418719

# Лучшее значение параметров  
clf\_gs.best\_params\_

{'n\_neighbors': 13}

# Изменение качества на тестовой выборке в зависимости от К-соседей  
plt.plot(n\_range, clf\_gs.cv\_results\_['mean\_test\_score'])

[<matplotlib.lines.Line2D at 0x7fdfa8e4b4c0>]



#### Randomized Search

%%time  
clf\_rs = RandomizedSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')  
clf\_rs.fit(x\_train, y\_train)

CPU times: user 175 ms, sys: 4.61 ms, total: 180 ms  
Wall time: 178 ms

RandomizedSearchCV(cv=5, estimator=KNeighborsClassifier(),  
 param\_distributions=[{'n\_neighbors': array([ 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,  
 22, 23, 24, 25, 26, 27, 28, 29, 30])}],  
 scoring='accuracy')

# В данном случае оба способа нашли одинаковое решение  
clf\_rs.best\_score\_, clf\_rs.best\_params\_

(0.9150246305418719, {'n\_neighbors': 13})

#### K-fold

X = data  
kf = KFold(n\_splits=10)  
for train, test in kf.split(X):  
 print("%s %s" % (train, test))

[ 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [ 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161  
 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177] [144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160]  
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35  
 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  
 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71  
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89  
 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107  
 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125  
 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143  
 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160] [161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177]

kf = KFold(n\_splits=10)  
scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=13),   
 x, y, scoring='accuracy',   
 cv=kf)  
scores

array([1. , 0.94444444, 0.88888889, 0.77777778, 0.77777778,  
 1. , 0.88888889, 0.44444444, 1. , 0.82352941])

kf = KFold(n\_splits=10)  
scores = cross\_validate(KNeighborsClassifier(n\_neighbors=13),   
 x, y, scoring='accuracy',   
 cv=kf, return\_train\_score=True)  
scores

{'fit\_time': array([0.00158596, 0.00122786, 0.00096488, 0.00090909, 0.000911 ,  
 0.00097299, 0.00092697, 0.00091195, 0.000916 , 0.00106096]),  
 'score\_time': array([0.00251007, 0.00188398, 0.00130892, 0.00129294, 0.00128102,  
 0.00133204, 0.00139809, 0.00127816, 0.00128102, 0.00137115]),  
 'test\_score': array([1. , 0.94444444, 0.88888889, 0.77777778, 0.77777778,  
 1. , 0.88888889, 0.44444444, 1. , 0.82352941]),  
 'train\_score': array([0.88125 , 0.89375 , 0.90625 , 0.9125 , 0.90625 ,  
 0.9 , 0.9125 , 0.925 , 0.8757764 , 0.88819876])}

#### ShuffleSplit

X = data  
# Эквивалент KFold(n\_splits=n)  
kf = ShuffleSplit(n\_splits=10, test\_size=0.25)  
for train, test in kf.split(X):  
 print("%s %s" % (train, test))

[ 80 52 159 25 169 2 177 26 89 83 34 66 72 98 161 45 113 38  
 5 160 94 41 103 99 28 37 127 13 55 114 50 48 145 174 147 117  
 156 79 39 6 33 40 155 67 1 101 137 110 76 168 132 14 151 176  
 32 68 116 36 65 51 9 75 81 108 107 47 43 27 53 102 70 130  
 171 7 64 44 128 16 141 166 11 0 15 153 122 73 105 163 170 35  
 144 63 21 92 164 85 17 119 96 112 62 42 109 87 19 123 133 18  
 8 3 167 118 162 57 125 139 143 22 124 100 56 82 23 149 115 60  
 20 148 29 135 88 58 136] [106 93 173 154 157 175 104 12 74 111 49 126 95 54 86 129 30 150  
 142 140 46 61 91 77 69 172 78 165 120 59 121 84 158 4 134 10  
 31 138 90 146 24 131 152 97 71]  
[140 19 83 91 60 67 102 174 81 28 11 69 53 131 77 1 114 23  
 33 2 88 159 117 71 55 137 130 95 133 62 172 45 20 37 30 61  
 51 82 21 27 169 42 52 123 136 16 87 170 165 148 101 73 121 151  
 153 50 163 132 35 168 24 120 32 138 128 6 57 15 107 0 96 65  
 143 89 106 105 108 145 141 76 122 171 25 75 9 166 104 154 150 10  
 34 4 46 63 38 142 36 116 18 85 39 160 125 97 118 84 90 68  
 8 59 64 7 111 161 156 157 177 47 155 14 54 173 167 56 134 113  
 109 110 103 139 40 149 162] [135 98 176 17 152 94 66 5 79 164 3 93 119 26 147 92 31 112  
 72 144 12 78 86 127 158 100 41 70 99 80 126 115 129 74 43 13  
 124 175 58 146 22 48 49 44 29]  
[111 151 112 72 30 134 131 45 96 19 26 117 4 89 68 62 172 162  
 139 64 58 174 46 14 120 59 53 142 107 87 154 101 105 34 80 163  
 173 23 137 74 116 92 153 55 51 145 119 135 20 136 60 114 168 138  
 82 91 90 67 126 10 175 125 104 84 13 39 146 56 106 158 17 5  
 83 159 43 129 164 27 52 54 94 63 81 69 6 79 157 71 86 66  
 1 100 32 141 28 143 31 49 144 75 93 165 70 65 85 122 33 176  
 133 40 123 8 98 128 140 41 124 127 61 48 0 38 35 115 110 102  
 152 44 12 109 99 16 36] [108 169 149 161 132 156 95 3 29 47 57 78 2 160 7 76 73 150  
 130 50 148 11 9 171 22 97 103 25 166 121 21 88 147 118 155 42  
 15 170 113 177 18 167 24 77 37]  
[165 13 71 173 67 111 38 141 100 90 102 156 62 130 66 126 163 167  
 60 118 89 33 88 162 176 15 124 70 122 31 150 19 35 76 43 132  
 82 50 147 63 92 157 61 149 133 153 6 79 127 52 108 14 8 77  
 40 140 10 29 20 106 86 42 5 135 47 139 104 131 45 121 158 113  
 170 83 112 137 44 105 84 37 57 115 69 123 74 128 51 78 75 129  
 34 99 46 161 93 73 109 32 9 168 146 18 64 30 98 119 7 97  
 138 59 28 120 3 87 24 56 27 103 114 16 94 159 36 117 26 39  
 23 21 49 136 91 95 169] [ 22 65 85 160 41 0 175 154 81 164 145 171 166 155 151 72 48 55  
 68 2 172 101 116 53 58 25 11 143 107 148 54 174 12 1 125 110  
 177 142 152 80 144 4 17 96 134]  
[115 33 11 59 88 2 136 103 31 13 167 157 105 61 57 93 16 155  
 171 128 3 14 19 74 49 55 70 130 65 142 64 169 44 118 71 66  
 90 175 89 94 40 147 58 91 96 48 166 1 80 10 38 172 101 112  
 26 45 110 154 148 140 69 87 21 54 150 41 149 67 111 77 43 159  
 117 145 68 83 79 102 60 151 139 12 23 63 144 78 168 98 122 109  
 22 15 7 34 62 132 17 97 6 76 56 84 120 134 173 121 119 42  
 46 29 138 162 5 129 20 32 51 176 108 4 127 125 47 152 92 86  
 36 170 135 75 85 137 72] [114 160 141 50 104 0 25 73 100 126 82 123 158 28 52 39 146 156  
 37 165 107 113 30 177 164 106 143 153 95 24 124 8 174 81 161 9  
 27 163 99 116 53 131 133 35 18]  
[ 32 131 166 36 148 71 96 164 103 16 74 63 124 156 143 105 87 135  
 45 2 97 57 161 1 8 29 121 14 170 174 139 30 75 5 117 7  
 155 27 28 152 137 73 104 70 15 37 141 26 167 51 39 24 132 110  
 90 76 146 10 92 12 67 61 149 40 118 136 81 4 99 13 35 171  
 172 102 42 113 112 158 126 111 134 43 9 52 88 49 56 6 47 33  
 58 150 163 3 109 64 153 95 157 80 38 127 142 59 44 119 140 54  
 125 53 46 107 86 19 50 94 115 123 72 65 89 106 159 82 18 175  
 77 11 84 48 147 23 160] [ 55 22 130 122 114 41 173 17 20 154 25 62 66 151 177 162 83 145  
 85 79 120 98 101 116 108 91 34 21 100 68 169 138 165 78 93 144  
 69 129 176 60 133 31 168 0 128]  
[128 136 88 170 155 27 65 124 80 89 104 123 59 62 107 7 161 162  
 36 110 91 45 94 151 111 44 139 69 142 99 87 5 37 173 100 146  
 21 50 163 120 137 127 174 55 67 23 101 12 81 63 105 71 78 24  
 82 103 93 72 141 77 39 66 48 83 131 58 68 60 53 40 130 35  
 17 167 79 154 164 14 47 168 74 159 22 84 97 73 56 140 149 166  
 176 10 109 38 145 19 144 32 122 135 171 156 18 61 138 34 11 29  
 52 57 64 153 134 157 4 15 13 115 126 20 33 125 54 129 9 25  
 150 0 51 43 160 114 172] [ 90 143 116 119 133 76 113 3 108 96 169 70 158 175 177 147 132 28  
 95 85 152 112 8 49 102 106 2 118 16 75 86 121 26 117 165 92  
 98 148 42 6 1 30 31 46 41]  
[ 82 155 91 48 12 3 77 124 10 54 18 40 26 46 50 164 80 141  
 172 154 22 159 70 166 150 149 99 121 21 109 51 20 116 35 151 174  
 84 55 146 4 47 19 7 131 63 9 157 44 5 88 2 81 17 72  
 139 113 8 117 152 162 156 30 134 142 107 56 112 36 89 49 97 168  
 111 61 140 24 38 143 163 158 110 79 167 57 161 169 102 58 71 160  
 32 118 76 170 153 83 78 60 25 23 177 85 126 123 62 95 67 53  
 6 13 132 33 73 144 59 125 104 114 127 133 100 106 137 90 11 101  
 171 145 0 136 41 176 108] [ 86 93 16 129 15 28 98 45 115 74 34 96 165 128 103 14 42 119  
 147 92 27 135 94 122 105 65 43 66 52 173 75 39 130 68 64 175  
 120 87 148 37 138 1 69 31 29]  
[100 105 103 131 114 136 61 42 155 123 108 76 82 10 60 88 6 71  
 15 106 154 2 75 63 65 177 44 116 92 126 167 31 17 50 80 68  
 26 73 30 112 27 85 56 144 141 124 147 67 24 19 117 9 162 164  
 98 151 21 150 120 104 139 36 86 130 122 115 95 28 43 8 97 78  
 133 148 134 140 113 145 149 135 79 54 38 121 39 166 118 107 163 160  
 34 46 128 58 1 23 109 157 64 13 172 20 87 91 12 25 102 170  
 137 35 37 55 99 45 153 168 143 138 158 132 173 169 62 119 175 174  
 111 93 3 77 18 49 83] [161 51 40 127 11 33 14 5 66 4 152 176 41 165 70 129 84 29  
 89 48 159 142 74 52 90 125 72 156 7 110 94 81 0 22 57 47  
 59 146 53 16 69 32 96 171 101]  
[ 57 124 168 48 117 141 138 12 123 24 23 21 77 33 89 119 101 132  
 50 175 55 158 172 167 166 146 103 92 70 5 43 118 171 1 40 39  
 134 17 54 99 139 100 68 108 113 46 79 98 27 96 34 85 127 150  
 6 121 128 125 84 94 177 75 112 130 19 97 170 106 58 147 2 152  
 4 109 88 87 32 13 72 14 149 95 129 36 10 110 56 74 154 59  
 90 148 164 143 105 114 163 80 73 71 18 15 115 63 102 20 52 144  
 37 162 116 120 29 107 122 44 67 173 7 60 26 140 35 157 82 86  
 22 30 16 174 142 61 49] [ 38 131 91 47 8 66 135 126 65 153 151 42 136 31 176 45 25 62  
 169 69 155 145 51 78 165 11 137 81 93 53 64 83 156 161 9 0  
 111 76 104 3 159 133 28 160 41]

kf = ShuffleSplit(n\_splits=10, test\_size=0.25)  
scores = cross\_val\_score(KNeighborsClassifier(n\_neighbors=13),   
 x, y, scoring='accuracy',   
 cv=kf)  
scores

array([0.91111111, 0.86666667, 0.91111111, 0.86666667, 0.84444444,  
 0.91111111, 0.88888889, 0.86666667, 0.91111111, 0.93333333])

kf = ShuffleSplit(n\_splits=10, test\_size=0.25)  
scores = cross\_validate(KNeighborsClassifier(n\_neighbors=13),   
 x, y, scoring='accuracy',   
 cv=kf, return\_train\_score=True)  
scores

{'fit\_time': array([0.00158906, 0.00115108, 0.00091195, 0.00154209, 0.00096297,  
 0.00114989, 0.00096989, 0.0009079 , 0.00104809, 0.00095606]),  
 'score\_time': array([0.00360298, 0.00205302, 0.00184298, 0.00197792, 0.00187302,  
 0.00240898, 0.00179625, 0.00195622, 0.00184774, 0.00179505]),  
 'test\_score': array([0.93333333, 0.88888889, 0.91111111, 0.84444444, 0.91111111,  
 0.91111111, 0.82222222, 0.86666667, 0.93333333, 0.86666667]),  
 'train\_score': array([0.87969925, 0.89473684, 0.89473684, 0.90225564, 0.87218045,  
 0.90225564, 0.90977444, 0.89473684, 0.88721805, 0.89473684])}