Вибір змінних для побудови моделі

Tg @Heorhii_Kantsedal

Навіщо фільтрувати змінні ?

- Зменшення кількості шуму, а отже зниження ризиків перенавчання
- Побудова пояснювальних моделей закономірності котрих можуть використовуватись в бізнес процесах
- Використання моделей для предіктід котрол
- Зменшення необхідних ресурсів для навчання моделі
- Зменшення часу тренування, можливість перебрати більше гіпер параметрів моделі

Термінологія

- Змінна (feature) – властивість притаманна певному виміру/випадку для якого необхідно провести прогнозування/класифікацію/кластиризацію
- Цільова змінна (target) – власне результат прогнозування/класифікації/кластеризації. У випадку якщо таргет взяти з тестувальної вибірки може називатися прогнозом моделі.
- Тестова вибірка – вибірка яка не приймає участь в навчанні моделі, однак може бути викоривтана для підбіру гіпер мараметрів моделі (тобто дефакто приймає участь в навчанні опосередковано)
- Валідаційна вибірка – вибірка даних яка повністю не бере участь в тестуванні моделі. Точність на ній приймається за фінальну точність роботи моделі.
- Тренувальна вибірка – вибірка що бозпосередньо приймає участь в тренуванні.

Константні, квазі-константні та фічі, що дублюються

Константні фічі – це ті, які показують однакове значення, лише одне значення, для всіх спостережень з набору даних. Іншими словами, однакове значення для всіх рядків набору даних. Ці функції не надають інформації, яка дозволяє моделі машинного навчання розрізняти або передбачати таргет

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_selection import VarianceThreshold

data = pd.read_csv('../dataset_1.csv')
data.shape
```

```
(50000, 301)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    data.drop(labels=['target'], axis=1),  # drop the target
    data['target'],  # just the target
    test_size=0.3,
    random_state=0)


sel = VarianceThreshold(threshold=0)

sel.fit(X_train)

print("Не константні фічі", sum(sel.get_support()))
```

```
Не константні фічі 266
```

Квазіконстантні фічі

Квазіконстантні фічі – це ті, які показують однакове значення для переважної більшості спостережень набору даних. Загалом, ці функції надають мало інформації, якщо взагалі надають, Але можуть бути вийнятки. Тому ви повинні бути обережними, видаляючи ці типи функцій.

```
sel = VarianceThreshold(threshold=0.01)

sel.fit(X_train)

print("Не константні фічі і не квазі-константні фічі", sum(sel.get_support()))
```

```
Не константні фічі і не квазі-константні фічі 215
```

Фічі, що повторюються

Данні фічі вже мають дисперсію і не можуть бути виділені попередніми методами. Тим не манш вони не несуть інформацію і підлягають видаленню.

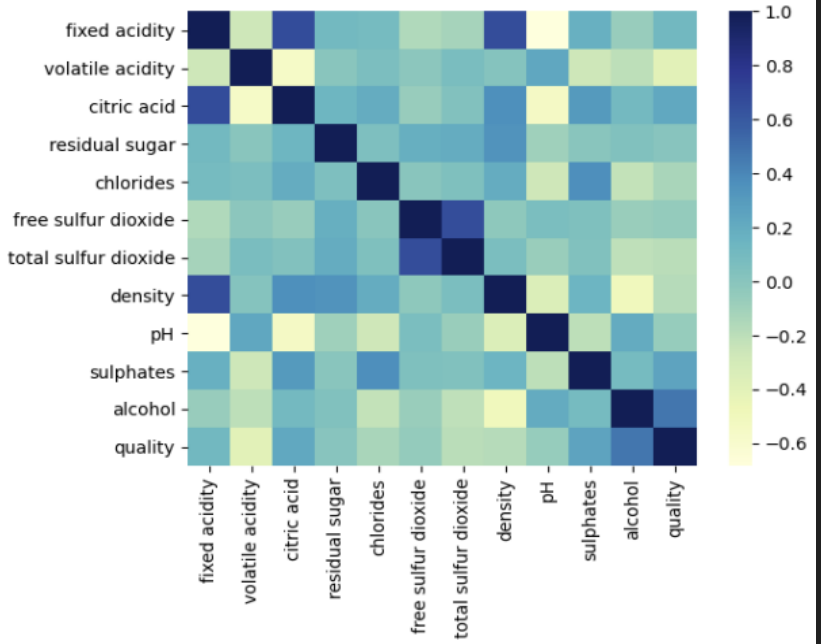
```
# check for duplicated features in the training set:

# create an empty dictionary, where we will store
# the groups of duplicates
duplicated_feat_pairs = {}

# create an empty list to collect features
# that were found to be duplicated
_duplicated_feat = []


# iterate over every feature in our dataset:
for i in range(0, len(X_train.columns)):
    # choose 1 feature:
    feat_1 = X_train.columns[i]

    # check if this feature has already been identified
    # as a duplicate of another one. If it was, it should be stored in
    # our _duplicated_feat list.

    # If this feature was already identified as a duplicate, we skip it, if
    # it has not yet been identified as a duplicate, then we proceed:
    if feat_1 not in _duplicated_feat:

        # create an empty list as an entry for this feature in the dictionary
        duplicated_feat_pairs[feat_1] = []

        # now, iterate over the remaining features of the dataset:
        for feat_2 in X_train.columns[i + 1:]:

            # check if this second feature is identical to the first one
            if X_train[feat_1].equals(X_train[feat_2]):

                # if it is identical, append it to the list in the dictionary
                duplicated_feat_pairs[feat_1].append(feat_2)

                # and append it to our monitor list for duplicated variables
                _duplicated_feat.append(feat_2)

                # done!


print("кількість фічей, що повторюються: ", len(_duplicated_feat))
```

```
кількість фічей, що повторюються:  6
```

Корреляційний аналіз. Кореляції Пірсона

Коефіцієнт кореляції Пірсона між двома змінними дорівнює коваріації двох змінних, або сумі добутків відхилень, поділеній на добуток їх стандартних відхилень

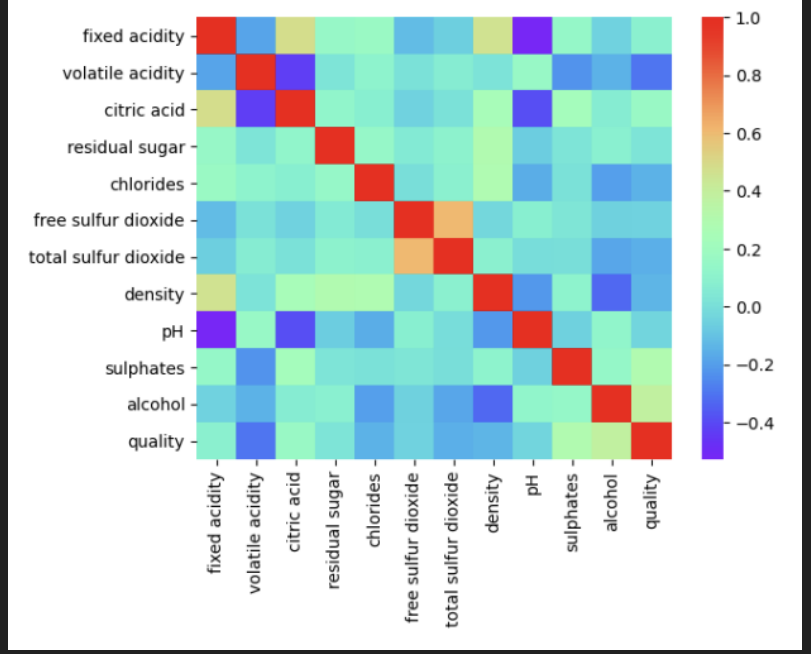
```
corrmat = data.corr(method='pearson')

# we can make a heatmap with the package seaborn
# and customise the colours of searborn's heatmap
cmap = "YlGnBu"

# and now plot the correlation matrix
sns.heatmap(corrmat, cmap=cmap)
plt.show()
```



Коефіцієнт кореляції рангу Кендала

- Коефіцієнт знаходить в діапазоні $-1 \leqslant \tau \leqslant 1$.
- Якщо узгодженість між двома величинами X та Y є ідеальною (тобто ранги двох величин збігаються), то коефіцієнт має значення 1.
- Якщо розбіжність між двома величинами X та Y є ідеальною (тобто вони мають обернені порядки зростання), то коефіцієнт дорівнює −1.
- Якщо X та Y незалежні, то математичне сподівання 0.

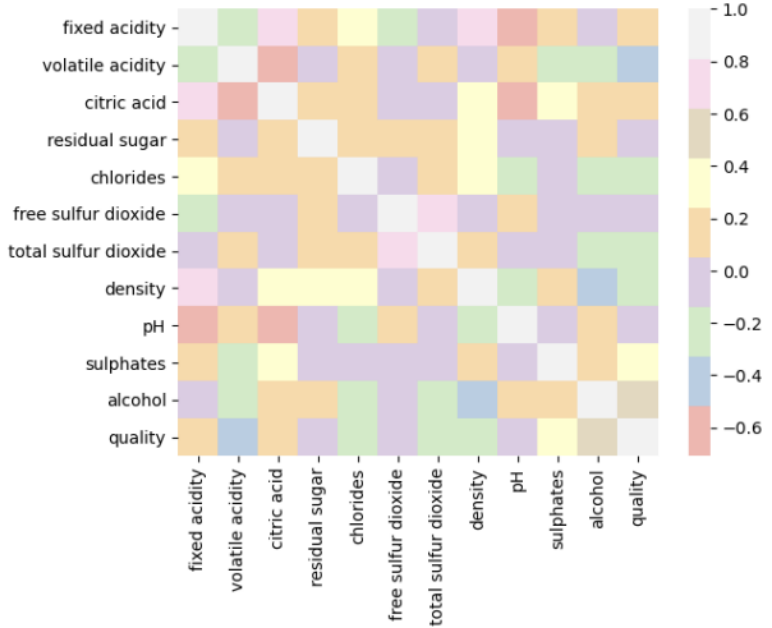
```
corrmat = data.corr(method='kendall')

# we can make a heatmap with the package seaborn
# and customise the colours of searborn's heatmap
cmap = "rainbow"

# and now plot the correlation matrix
sns.heatmap(corrmat, cmap=cmap)
plt.show()
```



Коефіцієнт кореляції рангу Спірмена

Коефіцієнт кореляції рангу Спірмена — непараметрична міра статистичної залежності між двома змінними; названий на честь Чарльза Спірмена. Він оцінює наскільки добре можна описати відношення між двома змінними за допомогою монотонної функції.

```
corrmat = data.corr(method='spearman')

# we can make a heatmap with the package seaborn
# and customise the colours of searborn's heatmap
cmap = "Pastel1"

# and now plot the correlation matrix
sns.heatmap(corrmat, cmap=cmap)
plt.show()
```



Статистичні тести. Взаємна інформація

Цей метод в основному використовує взаємну інформацію. Він обчислює взаємну інформаційну цінність для кожної з незалежних змінних щодо залежної змінної та вибирає ті, які мають найбільший приріст інформації. Іншими словами, він в основному вимірює залежність функцій від цільового значення. Вищий бал означає більше залежних змінних.

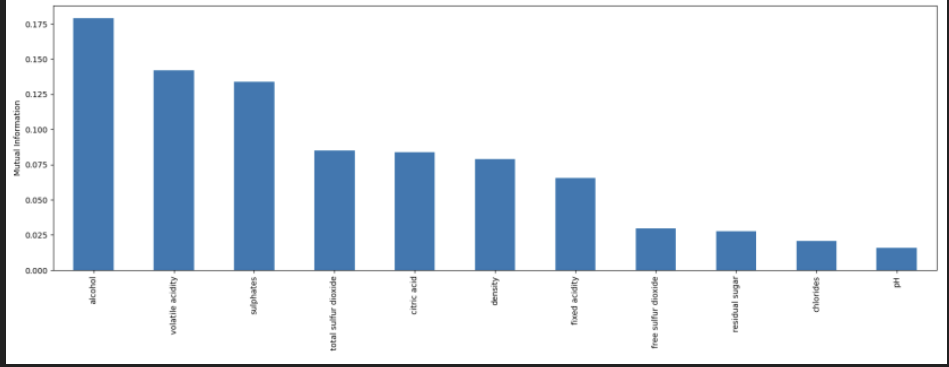
```
from sklearn.feature_selection import mutual_info_classif, mutual_info_regression
from sklearn.feature_selection import SelectKBest, SelectPercentile

X_train, X_test, y_train, y_test = train_test_split(
    data.drop(labels=['quality'], axis=1),
    data['quality'],
    test_size=0.3,
    random_state=0)

X_train.shape, X_test.shape
```

```
((1119, 11), (480, 11))
```

```
mi = mutual_info_classif(X_train, y_train)
mi
```

```
array([0.0656731 , 0.141957  , 0.08371092, 0.0275294 , 0.0205666 ,
       0.02972429, 0.08507245, 0.07901   , 0.01568985, 0.13370793,
       0.17908226])
```

```
mi = pd.Series(mi)
mi.index = X_train.columns
mi.sort_values(ascending=False).plot.bar(figsize=(20, 6))
plt.ylabel('Mutual Information')
plt.show()
```



Критерій узгодженості Пірсона

- Обчисліть статистику хі-квадрат між кожною невід'ємною ознакою та класом. Цей показник можна використовувати для вибору змінних із найвищими значеннями для тестової статистики хі-квадрат із X, яка має містити лише невід'ємні характеристики, такі як логічні значення або частоти (наприклад, кількість термінів у класифікації документів), відносно класи.
- Пам'ятайте, що тест хі-квадрат вимірює залежність між стохастичними змінними, тому використання цієї функції «відсіває» ознаки, які, швидше за все, будуть незалежними від класу і, отже, не релевантними для класифікації..

```
from sklearn.feature_selection import chi2

f_score = chi2(X_train.fillna(0), y_train)

# the 2 arrays of values
f_score
```

```
(array([9.78644174e+00, 1.39589843e+01, 1.12506164e+01, 2.53963967e+00,
        5.27188536e-01, 1.41209200e+02, 2.25343490e+03, 1.53887317e-04,
        1.39435866e-01, 3.48872790e+00, 3.14040226e+01]),
 array([8.15174734e-02, 1.58720630e-02, 4.66312902e-02, 7.70513784e-01,
        9.91093899e-01, 9.89962151e-29, 0.00000000e+00, 1.00000000e+00,
        9.99632544e-01, 6.25093923e-01, 7.79458351e-06]))
```

```
pvalues = pd.Series(f_score[1])
pvalues.index = X_train.columns
pvalues.sort_values(ascending=True)
```

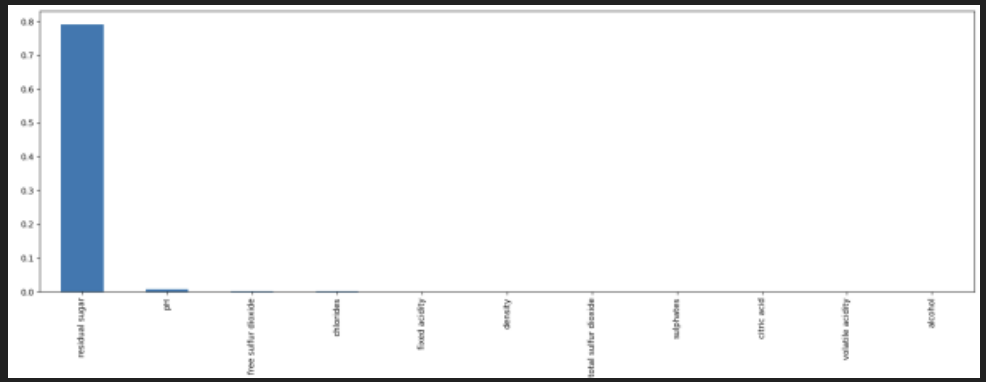
```
total sulfur dioxide    0.000000e+00
free sulfur dioxide     9.899622e-29
alcohol                 7.794584e-06
volatile acidity        1.587206e-02
citric acid             4.663129e-02
fixed acidity           8.151747e-02
sulphates               6.250939e-01
residual sugar          7.705138e-01
chlorides               9.910939e-01
pH                      9.996325e-01
density                 1.000000e+00
dtype: float64
```

```
sel_ = SelectKBest(chi2, k=3).fit(X_train, y_train)

# display features
X_train.columns[sel_.get_support()]
```

```
Index(['free sulfur dioxide', 'total sulfur dioxide', 'alcohol'], dtype='object')
```

Однофакторний вибір фіч

Вибір одновимірних фіч працює шляхом вибору найкращих ознак на основі одновимірних статистичних тестів (ANOVA). Методи оцінюють ступінь лінійної залежності між двома випадковими величинами. У цьому випадку будь-яка зі змінних-провісників і ціль. ANOVA передбачає лінійну залежність між об'єктом і метою, а також те, що змінні слідують гауссовому розподілу. Якщо це не так, результат цього тесту може бути некорисним. Це не завжди стосується змінних у вашому наборі даних, тому, якщо ви хочете реалізувати цю процедуру, вам потрібно буде підтвердити ці припущення.




```
from sklearn.feature_selection import f_classif, f_regression

univariate = f_classif(X_train, y_train)

univariate
```

```
(array([ 5.48808536, 53.09150257, 17.29450366,  0.58005007,  3.8081286 ,
         4.02817052, 19.74166577,  8.79657486,  3.88407232, 15.35291735,
        77.36045943]),
 array([5.34732355e-05, 1.69807093e-49, 1.63780294e-16, 7.15341143e-01,
        2.00203614e-03, 1.25642558e-03, 6.90373814e-19, 3.37520249e-08,
        1.70531063e-03, 1.28201477e-14, 1.06707132e-69]))
```

```
sel_ = SelectKBest(f_classif, k=10).fit(X_train, y_train)

# display selected feature names
X_train.columns[sel_.get_support()]
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'chlorides',
       'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH',
       'sulphates', 'alcohol'],
      dtype='object')
```

```
# univariate anova
univariate = f_regression(X_train.fillna(0), y_train)

# plot values
univariate = pd.Series(univariate[1])
univariate.index = X_train.columns
univariate.sort_values(ascending=False).plot.bar(figsize=(20,6))
plt.show()
```

Регрессія + Регуляризація

Регуляризація вводить штраф за включення зайвих областей функціонального простору, що використовується для побудови моделі і це може покращити узагальнення.

```
from sklearn.linear_model import Lasso, LogisticRegression
from sklearn.feature_selection import SelectFromModel
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

sel_ = SelectFromModel(
    LogisticRegression(C=0.5, penalty='l1', solver='liblinear', random_state=10))

sel_.fit(scaler.transform(X_train), y_train)

sel_.get_support()
```

```
/home/pollymorphism/miniconda3/lib/python3.7/site-packages/sklearn/linear_model/log
Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class opti
  "this warning.", FutureWarning)
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

```
selected_feat = X_train.columns[(sel_.get_support())]

print('total features: {}'.format((X_train.shape[1])))
print('selected features: {}'.format(len(selected_feat)))
```

```
total features: 11
selected features: 11
features with coefficients shrank to zero: 18
```

End…