In [104…
```
import pandas as pd
import numpy as np
from numpy.random import seed

import warnings
warnings.filterwarnings("ignore")
```

In [104…
```
seed(100)
```

In [ ]:
```
```

In [104…
```
df = pd.read_csv("./DATA/online_shoppers_intention.csv")
df.head()
```

Out[104…

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelate
0	0.0	0.0	0.0	0.0	1
1	0.0	0.0	0.0	0.0	2
2	0.0	-1.0	0.0	-1.0	1
3	0.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	0.0	10

In [105…
```
df.shape
```

Out[105… (12330, 18)

In [105…
```
cols = df.columns.tolist()
cols
```

Out[105…
```
['Administrative',
 'Administrative_Duration',
 'Informational',
 'Informational_Duration',
 'ProductRelated',
 'ProductRelated_Duration',
 'BounceRates',
 'ExitRates',
 'PageValues',
 'SpecialDay',
 'Month',
 'OperatingSystems',
 'Browser',
 'Region',
 'TrafficType',
 'VisitorType',
 'Weekend',
 'Revenue']
```

In [105…
```
df = df[ df["Administrative_Duration"]>=0 ]
df = df[ df['Informational_Duration']>=0 ]
df = df[ df['ProductRelated_Duration']>=0 ]
```

In [105…
```
df.shape
```

Out[105… (12283, 18)

In [105…
```
12330 - 12283
```

Out[105… 47

In [105…
```
df["Administrative"].isnull().value_counts()
```

Out[105…
```
False    12283
Name: Administrative, dtype: int64
```

In [105…
```
for c in cols:
    print(df[c].isnull().value_counts())
```

```
False    12283
Name: Administrative, dtype: int64
False    12283
Name: Administrative_Duration, dtype: int64
False    12283
Name: Informational, dtype: int64
False    12283
Name: Informational_Duration, dtype: int64
False    12283
Name: ProductRelated, dtype: int64
False    12283
Name: ProductRelated_Duration, dtype: int64
False    12283
Name: BounceRates, dtype: int64
False    12283
Name: ExitRates, dtype: int64
False    12283
Name: PageValues, dtype: int64
False    12283
Name: SpecialDay, dtype: int64
False    12283
Name: Month, dtype: int64
False    12283
Name: OperatingSystems, dtype: int64
False    12283
Name: Browser, dtype: int64
False    12283
Name: Region, dtype: int64
False    12283
Name: TrafficType, dtype: int64
False    12283
Name: VisitorType, dtype: int64
False    12283
Name: Weekend, dtype: int64
False    12283
Name: Revenue, dtype: int64
```

In [105…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12283 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Administrative           12283 non-null  float64
 1   Administrative_Duration  12283 non-null  float64
 2   Informational            12283 non-null  float64
 3   Informational_Duration   12283 non-null  float64
 4   ProductRelated           12283 non-null  float64
 5   ProductRelated_Duration  12283 non-null  float64
 6   BounceRates              12283 non-null  float64
 7   ExitRates                12283 non-null  float64
 8   PageValues               12283 non-null  float64
 9   SpecialDay               12283 non-null  float64
 10  Month                    12283 non-null  object 
 11  OperatingSystems         12283 non-null  int64  
 12  Browser                  12283 non-null  int64  
 13  Region                   12283 non-null  int64  
 14  TrafficType              12283 non-null  int64  
 15  VisitorType              12283 non-null  object 
 16  Weekend                  12283 non-null  bool   
 17  Revenue                  12283 non-null  bool   
dtypes: bool(2), float64(10), int64(4), object(2)
memory usage: 1.6+ MB
```

In [105…

```
m = df.Month.value_counts().index
mv = np.arange(1, len(m)+1,1).tolist()
df.Month.replace(to_replace=m, value=mv, inplace=True)
```

In [105…

```
vt = df.VisitorType.value_counts().index.tolist()
vtv = [1,2,3]
df.VisitorType.replace(to_replace=vt, value=vtv, inplace=True)
```

In [106…

```
w = df.Weekend.value_counts().index.tolist()
wv = [1,2]
df.Weekend.replace(to_replace=w, value=wv, inplace=True)
```

In [106…

```
r = df.Revenue.value_counts().index.tolist()
rv = [1,2]
df.Revenue.replace(to_replace=r, value=rv, inplace=True)
```

In [106…

```
df.head()
```

Out[106…

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelate
0	0.0	0.0	0.0	0.0	1
1	0.0	0.0	0.0	0.0	2
3	0.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	0.0	10
5	0.0	0.0	0.0	0.0	19

In [106…

```
y = df.Revenue.values
y.shape
```

Out[106… (12283,)

In [ ]:

preprocessing.StandardScaler()

In [106…

```
from sklearn import preprocessing
```

In [106…

```
df1 = df.copy()
```

In [106…

```
df1.shape
```

Out[106… (12283, 18)

In [106…

```
cols1 = cols[0:len(cols)-1:1]
data = df[cols1]
data.shape
```

Out[106… (12283, 17)

In [106…

```
cols1
```

Out[106…
```
['Administrative',
 'Administrative_Duration',
 'Informational',
 'Informational_Duration',
 'ProductRelated',
 'ProductRelated_Duration',
 'BounceRates',
 'ExitRates',
 'PageValues',
 'SpecialDay',
 'Month',
 'OperatingSystems',
 'Browser',
 'Region',
 'TrafficType',
 'VisitorType',
 'Weekend']
```

In [106…
```
X = preprocessing.StandardScaler().fit(data).transform(data)
X.shape,
```

Out[106…
```
((12283, 17),)
```

In [107…
```
df[cols1]=X
df["Revenue"] = y
```

In [107…
```
df.head()
```

Out[107…

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelate
0	-0.698907	-0.458238	-0.397247	-0.245408	-0.69287
1	-0.698907	-0.458238	-0.397247	-0.245408	-0.6704
3	-0.698907	-0.458238	-0.397247	-0.245408	-0.6704
4	-0.698907	-0.458238	-0.397247	-0.245408	-0.4907
5	-0.698907	-0.458238	-0.397247	-0.245408	-0.28855

In [107…
```
df.describe()
```

Out[107…

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductR
count	1.228300e+04	1.228300e+04	1.228300e+04	1.228300e+04	1.22830
mean	4.574927e-15	1.315103e-15	3.743756e-15	4.117827e-15	1.2550
std	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00	1.00004
min	-6.989074e-01	-4.582377e-01	-3.972470e-01	-2.454083e-01	-7.1533
25%	-6.989074e-01	-4.582377e-01	-3.972470e-01	-2.454083e-01	-5.5810
50%	-3.981549e-01	-4.130501e-01	-3.972470e-01	-2.454083e-01	-3.1102
75%	5.041026e-01	7.243447e-02	-3.972470e-01	-2.454083e-01	1.3821
max	7.421410e+00	1.873944e+01	1.846939e+01	1.783568e+01	1.51203

In [107…

```
corr = df.corr(method="pearson")
corr.head()
```

Out[107…

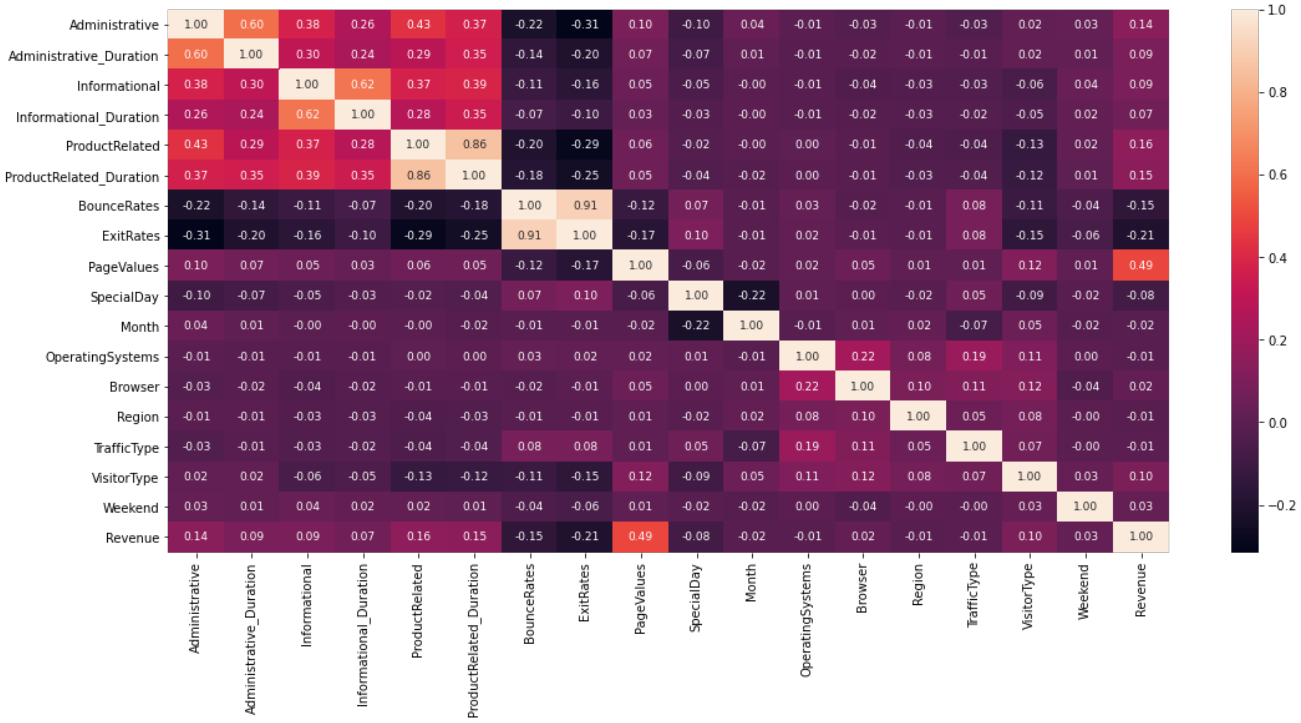
	Administrative	Administrative_Duration	Informational	Informational_D
Administrative	1.000000	0.601171	0.376374	0.
Administrative_Duration	0.601171	1.000000	0.302304	0.
Informational	0.376374	0.302304	1.000000	0.
Informational_Duration	0.255479	0.237734	0.618880	1.
ProductRelated	0.430116	0.288277	0.373680	0

In [107…

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [107…

```
plt.figure(figsize=(18,8))
sns.heatmap(corr, annot=True, annot_kws={"size":9}, fmt="0.2f")
plt.show()
```



In [107…

```
cor1 = corr.Revenue.values.tolist()
cor1 = cor1[0:len(cor1)-1:1]
cor1
```

Out[107…

```
[0.13796981861526525,
 0.09291149355015174,
 0.09467241450209762,
 0.06998354694413166,
 0.15762433876462859,
 0.15152794540687842,
 -0.14934778022367295,
 -0.20614547059815264,
 0.49231720815966273,
 -0.08263186998143683,
 -0.01563019306970006,
 -0.014841012469335188,
 0.024063467171549908,
 -0.011574735503028408,
 -0.005378089415888386,
 0.097892698361337,
 0.028936666543794204]
```

In [107…

```
cors = np.sort(cor1).tolist()
cors = cors[-1::-1]
cors
```

Out[107…
```
[0.49231720815966273,
 0.15762433876462859,
 0.15152794540687842,
 0.13796981861526525,
 0.097892698361337,
 0.09467241450209762,
 0.09291149355015174,
 0.06998354694413166,
 0.028936666543794204,
 0.024063467171549908,
 -0.005378089415888386,
 -0.011574735503028408,
 -0.014841012469335188,
 -0.01563019306970006,
 -0.08263186998143683,
 -0.14934778022367295,
 -0.20614547059815264]
```

In [107…
```
idx = np.argsort(cor1).tolist()
idx = idx[-1::-1]
idx
```

Out[107… `[8, 4, 5, 0, 15, 2, 1, 3, 16, 12, 14, 13, 11, 10, 9, 6, 7]`

In [107…
```
c_attr = []
for i in idx:
    c_attr.append(cols1[i])

c_attr
```

Out[107…
```
['PageValues',
 'ProductRelated',
 'ProductRelated_Duration',
 'Administrative',
 'VisitorType',
 'Informational',
 'Administrative_Duration',
 'Informational_Duration',
 'Weekend',
 'Browser',
 'TrafficType',
 'Region',
 'OperatingSystems',
 'Month',
 'SpecialDay',
 'BounceRates',
 'ExitRates']
```

In [108…
```
len(c_attr), len(idx)
```

Out[108… `(17, 17)`

In [108…

```
cor_r = pd.DataFrame({"attr":c_attr})
cor_r["cor"] = cors
cor_r
```

Out[108…

	attr	cor
0	PageValues	0.492317
1	ProductRelated	0.157624
2	ProductRelated_Duration	0.151528
3	Administrative	0.137970
4	VisitorType	0.097893
5	Informational	0.094672
6	Administrative_Duration	0.092911
7	Informational_Duration	0.069984
8	Weekend	0.028937
9	Browser	0.024063
10	TrafficType	-0.005378
11	Region	-0.011575
12	OperatingSystems	-0.014841
13	Month	-0.015630
14	SpecialDay	-0.082632
15	BounceRates	-0.149348
16	ExitRates	-0.206145

In [108…

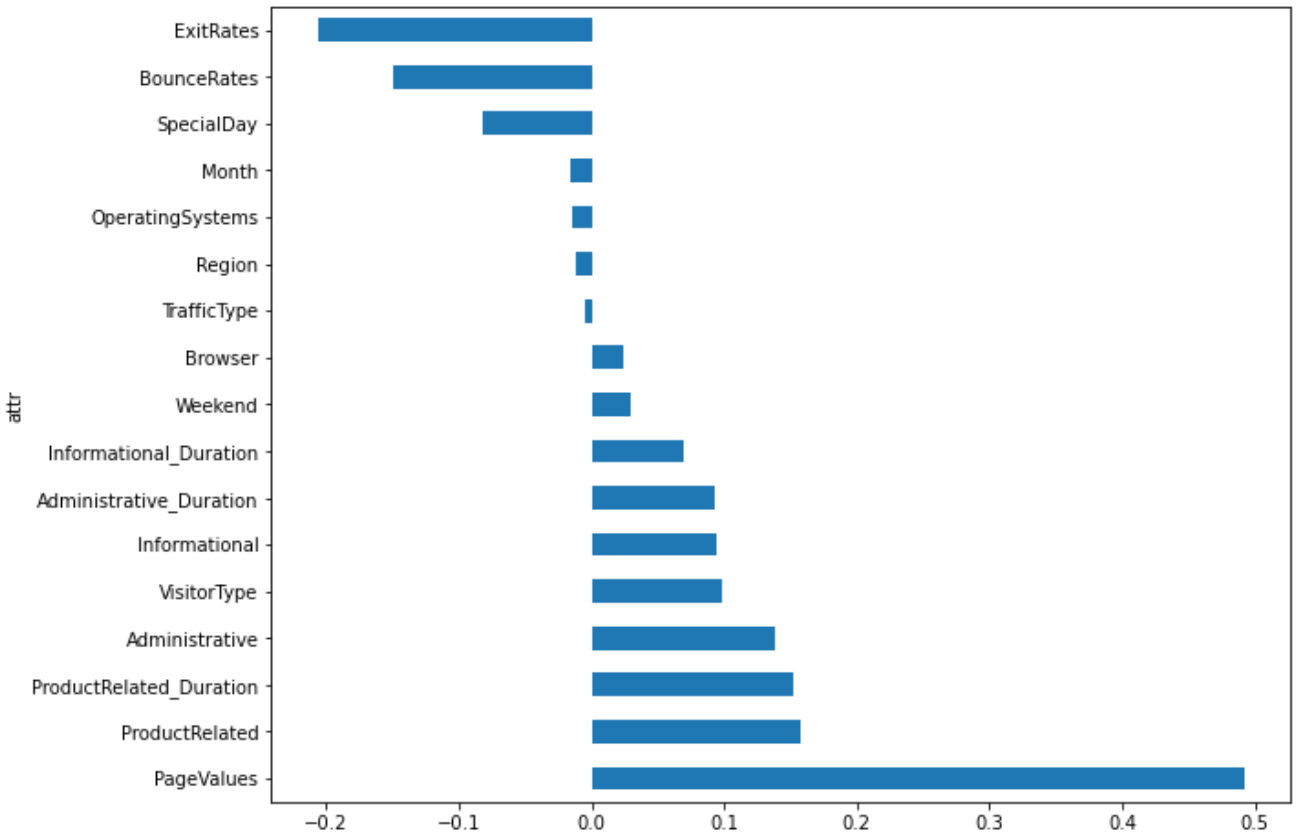
```
cor_r = cor_r.set_index(cor_r['attr'])
print(cor_r)
```

attr	attr	cor
PageValues	PageValues	0.492317
ProductRelated	ProductRelated	0.157624
ProductRelated_Duration	ProductRelated_Duration	0.151528
Administrative	Administrative	0.137970
VisitorType	VisitorType	0.097893
Informational	Informational	0.094672
Administrative_Duration	Administrative_Duration	0.092911
Informational_Duration	Informational_Duration	0.069984
Weekend	Weekend	0.028937
Browser	Browser	0.024063
TrafficType	TrafficType	-0.005378
Region	Region	-0.011575
OperatingSystems	OperatingSystems	-0.014841
Month	Month	-0.015630
SpecialDay	SpecialDay	-0.082632
BounceRates	BounceRates	-0.149348
ExitRates	ExitRates	-0.206145

In [108…

```
plt.figure(figsize=(10,8))
cor_r['cor'].plot(kind='barh')
```

Out[108… <AxesSubplot:ylabel='attr'>



Attribute

In [108…
```
c_attr
```

Out[108…
```
['PageValues',
 'ProductRelated',
 'ProductRelated_Duration',
 'Administrative',
 'VisitorType',
 'Informational',
 'Administrative_Duration',
 'Informational_Duration',
 'Weekend',
 'Browser',
 'TrafficType',
 'Region',
 'OperatingSystems',
 'Month',
 'SpecialDay',
 'BounceRates',
 'ExitRates']
```

In [ ]:
```

```

In [108…
```
cset1 = c_attr[0:5]
cset1
```

Out[108…
```
['PageValues',
 'ProductRelated',
 'ProductRelated_Duration',
 'Administrative',
 'VisitorType']
```

In [108…
```
c_attr1 = c_attr[-1::-1]
cset2 = c_attr1[0:3]
cset2
```

Out[108…
```
['ExitRates', 'BounceRates', 'SpecialDay']
```

In [108…
```
cols_s = cset1+cset2
cols_s
```

Out[108…
```
['PageValues',
 'ProductRelated',
 'ProductRelated_Duration',
 'Administrative',
 'VisitorType',
 'ExitRates',
 'BounceRates',
 'SpecialDay']
```

In [ ]:

In [108…
```
df2 = df.copy()
```

In [108…
```
df = df[cols_s]
df["Revenue"] = y
df.head()
```

Out[108…

	PageValues	ProductRelated	ProductRelated_Duration	Administrative	VisitorType	ExitRate
0	-0.317845	-0.692875	-0.625962	-0.698907	-0.401915	3.28156
1	-0.317845	-0.670413	-0.592556	-0.698907	-0.401915	1.19656
3	-0.317845	-0.670413	-0.624570	-0.698907	-0.401915	2.03056
4	-0.317845	-0.490717	-0.298432	-0.698907	-0.401915	0.15406
5	-0.317845	-0.288559	-0.545467	-0.698907	-0.401915	-0.37632

In [109…
```
df.shape
```

Out[109… (12283, 9)

In [ ]:

Random Collection

In [109…
```
from collections import Counter
```

In [109…
```
n = len(df)
idx1 = np.arange(n).tolist()

np.shape(idx1)
```

Out[109… (12283,)

Change parameter

In [109…
```
r_c = np.random.choice(idx1, 1000)
r_c.shape
```

Out[109… (1000,)

In [109…
```
c = Counter(r_c)
c.most_common()[0:5]
```

Out[109… [(324, 3), (6523, 3), (802, 2), (4231, 2), (11766, 2)]

In [109…
```
cl = list(c)
c1 = Counter(cl)
c1.most_common()[0:5]
```

Out[109… [(5640, 1), (6936, 1), (8039, 1), (12119, 1), (79, 1)]

In [109…
```
cl[0:5]
```

Out[109… [5640, 6936, 8039, 12119, 79]

In [109…
```
np.shape(cl)
```

Out[109… (967,)

In [109…
```
clu = np.unique(cl)
np.shape(clu)
```

Out[109… (967,)

In [109…
```
df = df.iloc[cl, :]
df.shape
```

Out[109… (967, 9)

In [ ]:
```

```

ML

In [110…
```
from sklearn.model_selection import train_test_split, GridSearchCV
```

In [110…
```
X = df[cols_s].values
y = df["Revenue"].values
X.shape, y.shape
```

Out[110… ((967, 8), (967,))

In [110…
```
x_tr, x_t, y_tr, y_t = train_test_split(X, y, test_size=0.3, random_state=100
x_tr.shape, x_t.shape, y_tr.shape, y_t.shape
```

Out[110… ((676, 8), (291, 8), (676,), (291,))

In [110…
```
from sklearn import svm
from sklearn.svm import SVC
```

In [110…
```
clf = GridSearchCV(svm.SVC(gamma='auto'), {'C':[1,10,20], 'kernel':['rbf','li
```

In [110…
```
clf.fit(x_tr, y_tr)
res = pd.DataFrame(clf.cv_results_)
```

In [110…
```
res.columns
```

Out[110…
```
Index(['mean_fit_time', 'std_fit_time', 'mean_score_time', 'std_score_time',
       'param_C', 'param_kernel', 'params', 'split0_test_score',
       'split1_test_score', 'split2_test_score', 'split3_test_score',
       'split4_test_score', 'mean_test_score', 'std_test_score',
       'rank_test_score'],
      dtype='object')
```

In [110…
```
res[ ['param_C', 'param_kernel', 'mean_test_score'] ]
```

Out[110…

	param_C	param_kernel	mean_test_score
0	1	rbf	0.892015
1	1	linear	0.894946
2	10	rbf	0.896471
3	10	linear	0.894946
4	20	rbf	0.896460
5	20	linear	0.894946

In [110…
```
res['mean_test_score'].max()
```

Out[110… 0.8964705882352941

0.8934989648033126 -> (500) 0.8964705882352941 -> (1000) ** 0.8923228876717249 -> (2000)

In [ ]:

In [110…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

In [111…
```
from sklearn import ensemble
```

In [ ]:

GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0)

In [111…

```
model_params = {
    'svm':{
        'model':svm.SVC(gamma='auto'),
        'params':{
            'C':[1, 10, 20],
            'kernel':['rbf','linear']
        }
    },
    'random_forest':{
        'model':RandomForestClassifier(),
        'params':{
            'n_estimators':[1,5,10]
        }
    },
    'logistic_regression':{
        'model':LogisticRegression(solver='liblinear', multi_class='auto'),
        'params':{
            'C':[1,5,10]
        }
    },
    'gradient_boosting':{
        'model':ensemble.GradientBoostingClassifier(learning_rate=1.0, max_de
        'params':{
            'n_estimators':[100,200,300]
        }
    }

}
```

In [111…

```
scores = []

for model_name, mp in model_params.items():
    clf = GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=Fa
    clf.fit(x_tr, y_tr)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
```

In [111…

```
df = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
df
```

Out[111…

	model	best_score	best_params
0	svm	0.896471	{'C': 10, 'kernel': 'rbf'}
1	random_forest	0.886111	{'n_estimators': 5}
2	logistic_regression	0.884619	{'C': 5}
3	gradient_boosting	0.889063	{'n_estimators': 300}

In [ ]: