Рубежный контроль №2, Черников Анатолий РТ5-61Б

Вариант 21 Методы: Метод опорных векторов, Градиентный бустинг

Ввод [99]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.preprocessing import OrdinalEncoder
```

Ввод [100]:

```
data = pd.read_csv("formual_E_Raceresults.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1502 entries, 0 to 1501
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   SeasonName       1502 non-null   object
 1   RaceName         1502 non-null   object
 2   Pos              1502 non-null   int64 
 3   DriverNumber     1502 non-null   object
 4   DriverFirstName  1502 non-null   object
 5   DriverLastName   1502 non-null   object
 6   Team             1502 non-null   object
 7   Started          1502 non-null   object
 8   Best             1502 non-null   object
 9   Time             1502 non-null   object
 10  PtsPoints        1502 non-null   int64 
dtypes: int64(2), object(9)
memory usage: 129.2+ KB
```

Ввод [101]:

```
data.head()
```

Out[101]:

	SeasonName	RaceName	Pos	DriverNumber	DriverFirstName	DriverLastName	Team	S
0	Season 1 2014/15	Beijing E-Prix	1	#11	Lucas	Di Grassi	Audi Sport ABT Formula E Team	
1	Season 1 2014/15	Beijing E-Prix	2	#27	Franck	Montagny	Andretti Autosport Formula E Team	
2	Season 1 2014/15	Beijing E-Prix	3	#2	Sam	Bird	Virgin Racing Formula E Team	
3	Season 1 2014/15	Beijing E-Prix	4	#28	Charles	Pic	Andretti Autosport Formula E Team	
4	Season 1 2014/15	Beijing E-Prix	5	#5	Karun	Chandhok	Mahindra Racing Formula E Team	

Ввод [102]:

```
data.drop("Time", axis=1, inplace=True)
```

Ввод [103]:

```
data.head()
```

Out[103]:

	SeasonName	RaceName	Pos	DriverNumber	DriverFirstName	DriverLastName	Team	S
0	Season 1 2014/15	Beijing E-Prix	1	#11	Lucas	Di Grassi	Audi Sport ABT Formula E Team	
1	Season 1 2014/15	Beijing E-Prix	2	#27	Franck	Montagny	Andretti Autosport Formula E Team	
2	Season 1 2014/15	Beijing E-Prix	3	#2	Sam	Bird	Virgin Racing Formula E Team	
3	Season 1 2014/15	Beijing E-Prix	4	#28	Charles	Pic	Andretti Autosport Formula E Team	
4	Season 1 2014/15	Beijing E-Prix	5	#5	Karun	Chandhok	Mahindra Racing Formula E Team	

Ввод [104]:

```
def best_to_float(x):
    splitted = x.split(':')
    value = 0
    try:
        data = float(splitted[0]) * 60 + float(splitted[1])
    except:
        data = float(splitted[0]) * 60
    return data
```

Ввод [105]:

```
data.drop(data[data['Best'] == '-'].index, inplace=True)
data['Best'] = data['Best'].str.replace("FL", '', regex=False)
data['Best'] = data['Best'].apply(lambda x: best_to_float(x))
```

Ввод [106]:

```
data['Started'] = data['Started'].str.replace("P", '', regex=False)
data['Started'] = data['Started'].apply(lambda x: float(x))
```

Ввод [107]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1473 entries, 0 to 1500
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   SeasonName       1473 non-null   object 
 1   RaceName         1473 non-null   object 
 2   Pos              1473 non-null   int64  
 3   DriverNumber     1473 non-null   object 
 4   DriverFirstName  1473 non-null   object 
 5   DriverLastName   1473 non-null   object 
 6   Team             1473 non-null   object 
 7   Started          1473 non-null   float64
 8   Best             1473 non-null   float64
 9   PtsPoints        1473 non-null   int64  
dtypes: float64(2), int64(2), object(6)
memory usage: 126.6+ KB
```

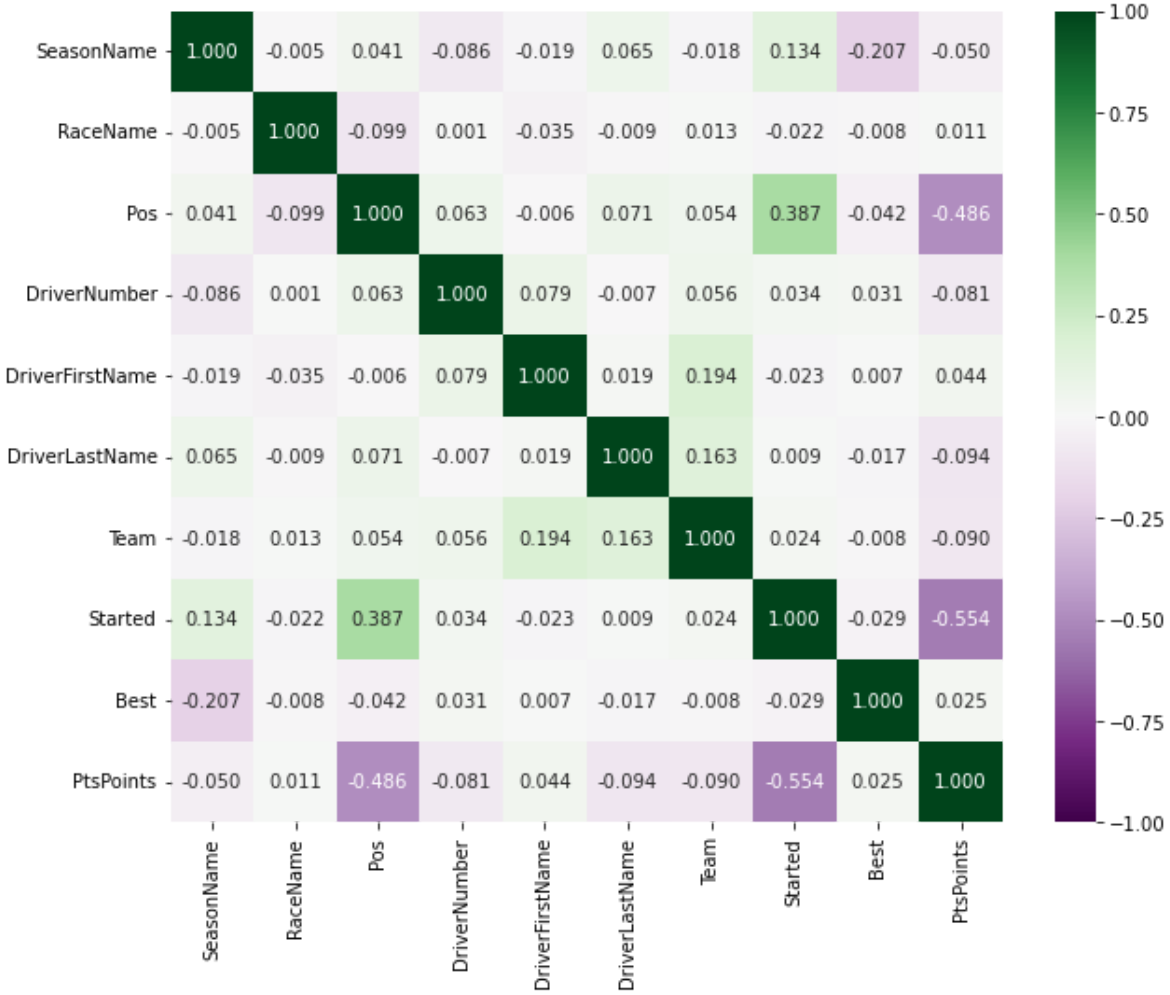
Ввод [108]:

```
oe = OrdinalEncoder()
oe_cols = oe.fit_transform(
    data[['DriverFirstName', 'DriverLastName', 'DriverNumber', 'RaceName', 'SeasonName', 'T

data[['DriverFirstName', 'DriverLastName', 'DriverNumber', 'RaceName', 'SeasonName', 'Team'

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1473 entries, 0 to 1500
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   SeasonName       1473 non-null   float64
 1   RaceName         1473 non-null   float64
 2   Pos              1473 non-null   int64  
 3   DriverNumber     1473 non-null   float64
 4   DriverFirstName  1473 non-null   float64
 5   DriverLastName   1473 non-null   float64
 6   Team             1473 non-null   float64
 7   Started          1473 non-null   float64
 8   Best             1473 non-null   float64
 9   PtsPoints        1473 non-null   int64  
dtypes: float64(8), int64(2)
memory usage: 126.6 KB
```

Ввод [109]:

```
plt.figure(figsize=(10,8))
sns.heatmap(data=data.corr(), annot=True, vmin=-1, vmax=1,  fmt='.3f', cmap='PRGn')
plt.show()
```



Ввод [111]:

```
from sklearn.model_selection import train_test_split, GridSearchCV

x = data.drop(columns=['PtsPoints'])
y = data['PtsPoints']
x_train : pd.DataFrame
x_test : pd.DataFrame
y_train : pd.Series
y_test : pd.Series

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=3)
```

Ввод [112]:

```
# Гиперпараметры для оптимизации
parameters_to_tune = {'max_depth' : np.arange(1, 6, 1), # 1-5
                      'min_samples_leaf' : np.linspace(0.02, 0.1, 5),
                      'max_features' : [0.2 , 0.4, 0.6, 0.8, 'auto', 'sqrt', 'log2']}
```

Ввод [127]:

```
from sklearn.tree import DecisionTreeRegressor, export_graphviz
# Решетчатый поиск со стратегией K-Fold
grid_search_reg= GridSearchCV(DecisionTreeRegressor(), parameters_to_tune, cv=KFold(n_split
grid_search_reg.fit(x, y)
```

Out[127]:

```
GridSearchCV(cv=KFold(n_splits=10, random_state=None, shuffle=False),
             estimator=DecisionTreeRegressor(),
             param_grid={'max_depth': array([1, 2, 3, 4, 5]),
                         'max_features': [0.2, 0.4, 0.6, 0.8, 'auto', 'sqr
t',
                                          'log2'],
                         'min_samples_leaf': array([0.02, 0.04, 0.06, 0.08,
0.1 ])},
             scoring='neg_root_mean_squared_error')
```

Ввод [115]:

```
grid_search_reg.best_params_
```

Out[115]:

```
{'max_depth': 5, 'max_features': 'auto', 'min_samples_leaf': 0.02}
```

Ввод [116]:

```
grid_search_reg.best_score_
```

Out[116]:

```
-0.7540148685969369
```

Ввод [117]:

```
# Обучение модели
dt_regressor : DecisionTreeRegressor = grid_search_reg.best_estimator_
dt_regressor.fit(x_train, y_train)
```

Out[117]:

```
DecisionTreeRegressor(max_depth=5, max_features='auto', min_samples_leaf=0.0
2)
```

Ввод [118]:

```
dt_pred = dt_regressor.predict(x_test)
```

Ввод [ ]:

```
from io import StringIO
from IPython.display import Image
import pydotplus
import graphviz

# Визуализация дерева
def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param,
                    filled=True, rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()
```

Ввод [137]:

```
Image(get_png_tree(dt_regressor, x_train.columns), height='100%')
```

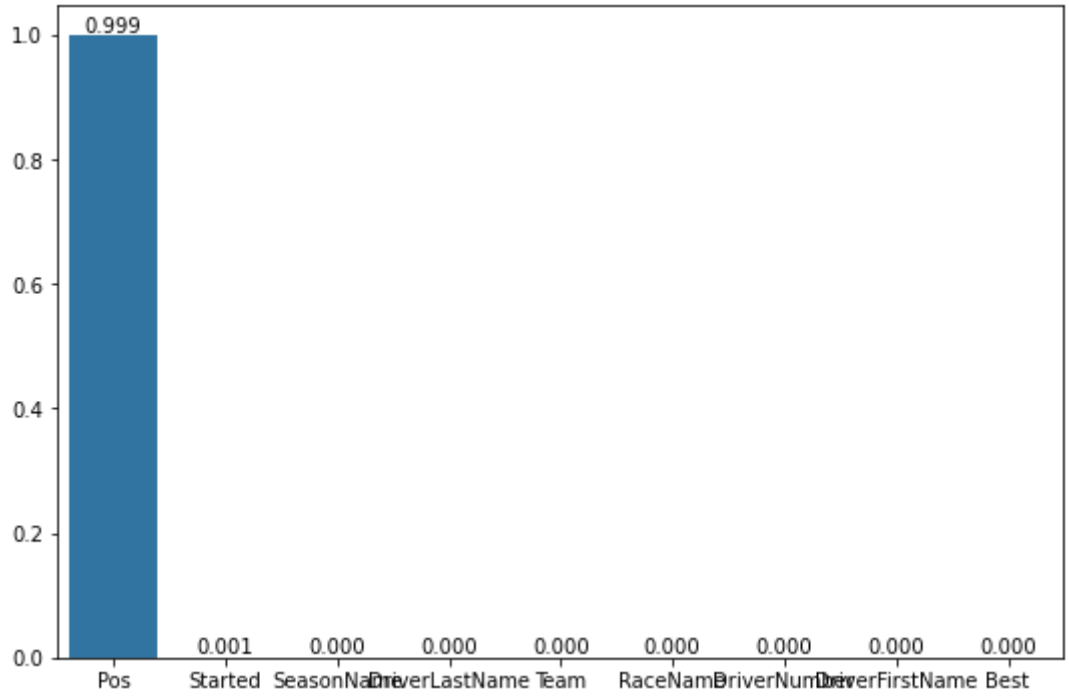
```
---------------------------------------------------------------------------
InvocationException                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13160/1553482478.py in <module>
----> 1 Image(get_png_tree(dt_regressor, x_train.columns), height='100%')

~\AppData\Local\Temp/ipykernel_13160/503620100.py in get_png_tree(tree_model
_param, feature_names_param)
     10                     filled=True, rounded=True, special_characters=Tr
ue)
     11     graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
---> 12     return graph.create_png()

~\anaconda3\lib\site-packages\pydotplus\graphviz.py in <lambda>(f, prog)
   1795             self.__setattr__(
   1796                 'create_' + frmt,
-> 1797                 lambda f=frmt, prog=self.prog: self.create(format=f,
prog=prog)
   1798             )
   1799             f = self.__dict__['create_' + frmt]

~\anaconda3\lib\site-packages\pydotplus\graphviz.py in create(self, prog, fo
rmat)
   1957             self.progs = find_graphviz()
   1958             if self.progs is None:
-> 1959                 raise InvocationException(
   1960                     'GraphViz\'s executables not found')
   1961

InvocationException: GraphViz's executables not found
```

Ввод [138]:

```
from operator import itemgetter

def plot_feature_importances(feature_names, feature_importances):
    """
    Функция визуализации важности признаков

    :param feature_names: Названия признаков
    :param feature_importances: Важности признаков
    """
    feature_importance_list = list(zip(feature_names, feature_importances))
    sorted_list = sorted(feature_importance_list, key=itemgetter(1), reverse=True)
    feature_order = [x for x, _ in sorted_list]

    plt.figure(figsize=(9,6))
    bar_plot = sns.barplot(x=feature_names, y=feature_importances, order=feature_order)
    bar_plot.bar_label(bar_plot.containers[-1], fmt='%.3f')
    plt.show()
```

Ввод [140]:

```
plot_feature_importances(x.columns.values, dt_regressor.feature_importances_)
```



Ввод [ ]:

```
# Гиперпараметры для оптимизации
parameters_to_tune = {'n_estimators' : [2, 5, 10],
                      'learning_rate': np.linspace(0.1, 0.3, 3),
                      'min_samples_split': np.arange(2, 5, 1), # 2-4
                      'max_depth' : np.arange(1, 5, 1)} # 1-4
```

Ввод [142]:

```
from sklearn.ensemble import GradientBoostingRegressor

gbc_gs = GridSearchCV(GradientBoostingRegressor(random_state=3),
                      parameters_to_tune, cv=5, scoring='neg_root_mean_squared_error')
gbc_gs.fit(x_train, y_train)
```

Out[142]:

```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(random_state=3),
             param_grid={'max_depth': array([1, 2, 3, 4, 5]),
                         'max_features': [0.2, 0.4, 0.6, 0.8, 'auto', 'sqr
t',
                                          'log2'],
                         'min_samples_leaf': array([0.02, 0.04, 0.06, 0.08,
0.1 ])},
             scoring='neg_root_mean_squared_error')
```

Ввод [143]:

```
gbc_gs.best_params_
```

Out[143]:

```
{'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 0.02}
```

Ввод [144]:

```
gbc_gs.best_score_
```

Out[144]:

```
-0.3467301898591439
```

Ввод [145]:

```
gb_regressor : GradientBoostingRegressor = gbc_gs.best_estimator_
gb_regressor.fit(x_train, y_train)
```

Out[145]:

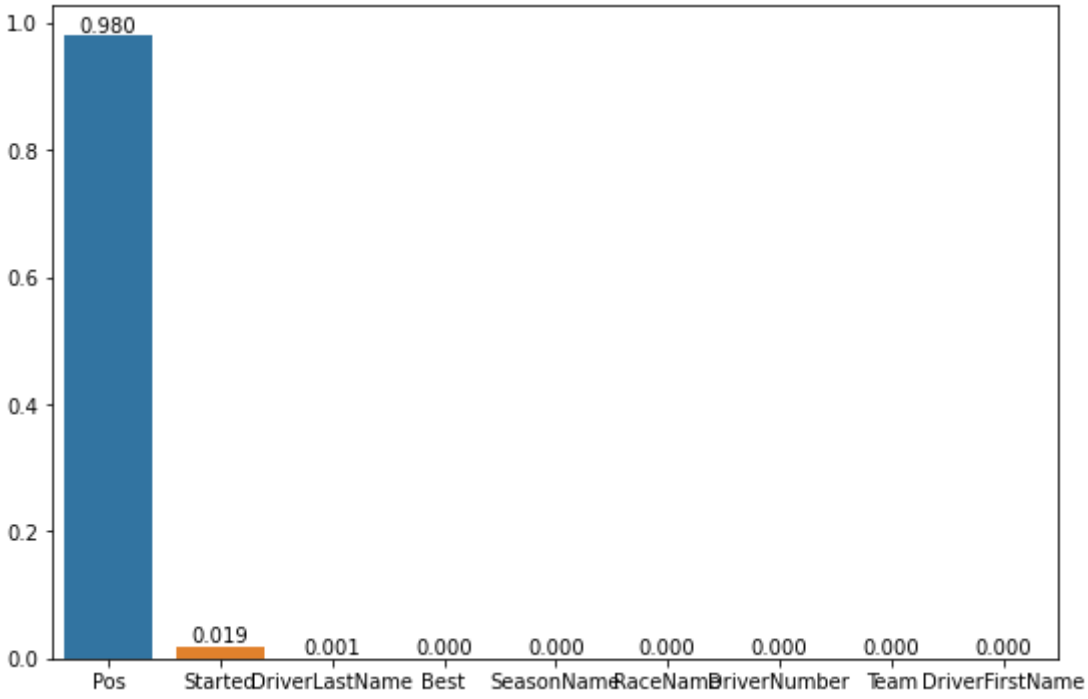
```
GradientBoostingRegressor(max_depth=4, max_features='auto',
                          min_samples_leaf=0.02, random_state=3)
```

Ввод [146]:

```
gb_pred = gb_regressor.predict(x_test)
```

Ввод [149]:

```
plot_feature_importances(x.columns.values, gb_regressor.feature_importances_)
```



Ввод [151]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error,

def print_regression_metrics(y_test, y_predicted):
    abs_err = mean_absolute_error(y_test, y_predicted)
    med_abs_err = median_absolute_error(y_test, y_predicted)
    mean_sq_err = mean_squared_error(y_test, y_predicted, squared=False)
    r2 = r2_score(y_test, y_predicted)

    return f"-Средняя абсолютная ошибка = {abs_err};\
          -Медианная абсолютная ошибка = {med_abs_err};\
          \-Среднеквадратичная ошибка = {mean_sq_err};\
          -Коэффициент детерминации = {r2}."
```

Ввод [153]:

```
print_regression_metrics(y_test, gb_pred)
```

Out[153]:

```
'-Средняя абсолютная ошибка = 0.20237178928963367;          -Медианная абсол
ютная ошибка = 0.06357408342386198;          \\-Среднеквадратичная ошибка =
0.41553948365840204;          -Коэффициент детерминации = 0.997201952809812
2.'
```

Ввод [154]:

```
print_regression_metrics(y_test, dt_pred)
```

Out[154]:

```
'-Средняя абсолютная ошибка = 0.43202778557237076;          -Медианная абсол
ютная ошибка = 0.1;          \\-Среднеквадратичная ошибка = 0.81516000211530
81;          -Коэффициент детерминации = 0.9892324681094119.'
```