Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления» Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»
Отчет по рубежному контролю №2
«Методы построения моделей машинного обучения»
Вариант №11

Выполнил:
Студент(ка) группы ИУ5-65Б
Тазенков Иван Дмитриевич

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись:_____

Дата:_____

Москва, 2023 г.

In [1]:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

In [2]:

```
#Загрузка датасета
data = pd.read_csv("hotel_bookings.csv")
```

In [3]:

```
data = data.head(500)
data.head()
```

Out[3]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month
0	Resort Hotel	0	342	2015	July	27	1
1	Resort Hotel	0	737	2015	July	27	1
2	Resort Hotel	0	7	2015	July	27	1
3	Resort Hotel	0	13	2015	July	27	1
4	Resort Hotel	0	14	2015	July	27	1

5 rows × 32 columns

In [4]:

```
data.shape
```

Out[4]:

```
(500, 32)
```

In [5]:

```
#Предобработка данных
#Проверка типов данных
data.dtypes
```

Out[5]:

```
hotel                           object
is_canceled                      int64
lead_time                        int64
arrival_date_year                int64
arrival_date_month              object
arrival_date_week_number         int64
arrival_date_day_of_month        int64
stays_in_weekend_nights          int64
stays_in_week_nights             int64
adults                           int64
children                       float64
```

```
babies                            int64
meal                             object
country                          object
market_segment                   object
distribution_channel             object
is_repeated_guest                 int64
previous_cancellations            int64
previous_bookings_not_canceled    int64
reserved_room_type               object
assigned_room_type               object
booking_changes                   int64
deposit_type                     object
agent                           float64
company                         float64
days_in_waiting_list              int64
customer_type                    object
adr                             float64
required_car_parking_spaces       int64
total_of_special_requests         int64
reservation_status               object
reservation_status_date          object
dtype: object
```

In [6]:

```
#Проверка пустых значений
data.isnull().sum()
```

Out[6]:

```
hotel                             0
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
meal                              0
country                           1
market_segment                    0
distribution_channel              0
is_repeated_guest                 0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
assigned_room_type                0
booking_changes                   0
deposit_type                      0
agent                            45
company                         493
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
reservation_status                0
reservation_status_date           0
dtype: int64
```

In [7]:

```
data = data.drop(columns = 'company')
```

In [8]:

```
data['country'].unique()
```

Out[8]:

```
array(['PRT', 'GBR', 'USA', 'ESP', 'IRL', 'FRA', nan, 'ROU', 'NOR', 'OMN',
       'ARG', 'POL', 'DEU', 'BEL', 'CHE', 'CN', 'GRC', 'ITA', 'NLD',
       'DNK', 'RUS', 'SWE', 'AUS', 'EST', 'CZE', 'BRA', 'FIN', 'MOZ',
       'BWA'], dtype=object)
```

In [9]:

```
data['agent'].unique()
```

Out[9]:

```
array([ nan, 304., 240., 303.,  15., 241.,   8., 250., 115.,   5., 175.,
       134., 156., 243., 242.,   3., 105.,  40., 147., 306., 184.,  96.,
         2., 127.,  95., 146.,   9., 177.,   6., 143., 244., 149., 167.,
       300., 171., 305.,  67., 196., 152., 142.])
```

In [10]:

```
data['agent'] = data['agent'].fillna(0)
```

In [11]:

```
data['country'] = data['country'].fillna('Unknown')
```

In [12]:

```
data['children'] = data['agent'].dropna()
```

In [13]:

```
data.isnull().sum()
```

Out[13]:

```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        0
babies                          0
meal                            0
country                         0
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
deposit_type                    0
agent                           0
days_in_waiting_list            0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
reservation_status_date         0
dtype: int64
```

In [14]:

```
#Кодирование категориальных признаков
```

```
LE = LabelEncoder()
for col in data.columns:
    if data[col].dtype == "object":
        data[col] = LE.fit_transform(data[col])
```

In [15]:

```
#Проверка типов данных
data.dtypes
```

Out[15]:

```
hotel                               int64
is_canceled                         int64
lead_time                           int64
arrival_date_year                   int64
arrival_date_month                  int64
arrival_date_week_number            int64
arrival_date_day_of_month           int64
stays_in_weekend_nights             int64
stays_in_week_nights                int64
adults                              int64
children                          float64
babies                              int64
meal                                int64
country                             int64
market_segment                      int64
distribution_channel                int64
is_repeated_guest                   int64
previous_cancellations              int64
previous_bookings_not_canceled      int64
reserved_room_type                  int64
assigned_room_type                  int64
booking_changes                     int64
deposit_type                        int64
agent                             float64
days_in_waiting_list                int64
customer_type                       int64
adr                               float64
required_car_parking_spaces         int64
total_of_special_requests           int64
reservation_status                  int64
reservation_status_date             int64
dtype: object
```

In [16]:

```
data.head()
```

Out[16]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	s
0	0	0	342	2015	0	27	1	
1	0	0	737	2015	0	27	1	
2	0	0	7	2015	0	27	1	
3	0	0	13	2015	0	27	1	
4	0	0	14	2015	0	27	1	

5 rows × 31 columns

In [17]:

```
#Разделение выборки на обучающую и тестовую
# target = "arrival_date_week_number"
# xArray = data.drop(target, axis=1)
# yArray = data[target]
X_train, X_test, y_train, y_test = train_test_split(data, data['arrival_date_week_number
```

```
'], test_size=0.2, random_state=1)
```

Регрессия. Метод опорных векторов (SVM)

In [18]:

```
from sklearn import svm
SVMRegr = svm.SVR()
SVMRegr.fit(X_train, y_train)
SVMRegr.score(X_test, y_test)
SVMRegr_predict = SVMRegr.predict(X_test)
```

In [19]:

```
print(SVMRegr_predict)
```

```
[28.16170352 28.0871845  28.01516047 28.06762748 28.0821204  28.11506963
 28.0370398  28.02708872 28.08229462 28.11634992 28.12179015 28.01023844
 28.14400785 28.09875956 28.07478279 28.00271488 28.15271046 28.11016468
 28.04453941 28.13441626 28.10569141 28.10876816 28.13081629 28.14678263
 28.13142482 28.03524465 28.05655822 28.09055527 28.11441899 28.17393763
 28.07017395 28.11764489 28.04618595 28.13531195 28.13497657 28.10429204
 28.10141636 28.08066773 28.02555947 28.07441582 28.11831892 28.10365866
 28.11700272 28.04881191 28.08629416 28.03913636 28.02519673 28.1035789
 28.10034992 28.05428678 28.15925168 28.13703269 28.08066773 28.10365866
 28.1529176  28.11304847 28.11658922 28.05388547 28.07218969 28.05033846
 28.05889447 27.95263213 28.07983074 27.9571159  28.06240383 28.1269901
 28.14596523 28.01570194 28.05431372 28.09086374 28.07089205 28.14589055
 28.04428895 28.0395048  28.02903932 28.11338535 28.07773199 28.10352417
 28.0414113  28.07528375 28.13133878 27.99872499 28.13780222 28.10366852
 28.05945662 27.91839568 28.18030341 28.07674972 28.12991678 28.08029227
 28.06251268 28.06251268 28.10352417 28.052272   28.09659383 28.11331591
 28.03694128 28.08194162 27.96317872 28.05805515]
```

In [20]:

```
# Создание графика фактических значений и предсказанных значений
plt.scatter(y_test, SVMRegr_predict)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--', lw=2)
plt.xlabel("Фактические значения")
plt.ylabel("Предсказанные значения")
plt.title("Сравнение фактических и предсказанных значений SVM")
plt.show()
```

Регрессия. Градиентный бустинг.

In [21]:

```
from sklearn.datasets import make_regression
from sklearn.ensemble import GradientBoostingRegressor

X, y = make_regression(random_state=0)
GBR_reg = GradientBoostingRegressor(random_state=0)
GBR_reg.fit(X_train, y_train)
GBR_reg.score(X_test, y_test)
```

Out[21]:

```
0.9999999992744206
```

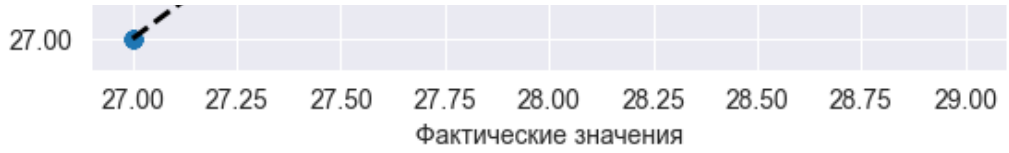
In [22]:

```
GBR_reg_predict = GBR_reg.predict(X_test)
print(GBR_reg_predict)
```

```
[28.00000259 28.99997603 27.00002915 27.00002915 28.99997603 28.99997603
 28.99997603 27.00002915 28.00000259 27.00002915 28.99997603 28.00000259
 27.00002915 27.00002915 28.00000259 28.00000259 28.99997603 28.00000259
 28.00000259 28.00000259 28.00000259 28.99997603 28.99997603 28.99997603
 27.00002915 28.99997603 27.00002915 28.99997603 28.00000259 28.00000259
 28.00000259 28.00000259 28.00000259 28.99997603 28.99997603 27.00002915
 28.00000259 27.00002915 28.99997603 28.99997603 28.99997603 28.99997603
 28.99997603 28.00000259 28.99997603 28.99997603 28.00000259 28.99997603
 28.99997603 27.00002915 28.00000259 28.00000259 27.00002915 28.99997603
 28.99997603 27.00002915 28.99997603 28.00000259 27.00002915 28.99997603
 28.99997603 27.00002915 28.00000259 28.99997603 28.99997603 28.99997603
 28.99997603 28.99997603 28.00000259 28.00000259 28.99997603 28.99997603
 28.00000259 27.00002915 27.00002915 28.99997603 28.99997603 28.99997603
 28.00000259 28.00000259 28.00000259 27.00002915 28.00000259 28.99997603
 28.00000259 28.00000259 28.99997603 28.00000259 28.99997603 28.99997603
 28.99997603 28.99997603 28.99997603 27.00002915 28.99997603 27.00002915
 28.00000259 27.00002915 28.00000259 27.00002915]
```

In [23]:

```
# Создание графика фактических значений и предсказанных значений
plt.scatter(y_test, GBR_reg_predict)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--', lw=2)
plt.xlabel("Фактические значения")
plt.ylabel("Предсказанные значения")
plt.title("Сравнение фактических и предсказанных значений GBR")
plt.show()
```



In [24]:

```
from sklearn.metrics import mean_squared_error, max_error, r2_score

X_train, X_test, y_train, y_test = train_test_split(data, data['arrival_date_week_number
'], test_size=0.3)

SVM_test = SVMRegr.predict(X_test)
SVM_MSE = mean_squared_error(y_pred=SVM_test, y_true=y_test)

SVM_test = SVMRegr.predict(X_test)
SVM_r2 = r2_score(y_pred=SVM_test, y_true=y_test)

GBR_test = GBR_reg.predict(X_test)
GBR_MSE = mean_squared_error(y_pred=GBR_test, y_true=y_test)

GBR_test = GBR_reg.predict(X_test)
GBR_r2 = r2_score(y_pred=GBR_test, y_true=y_test)

print(f"MSE:\n SVM: {SVM_MSE}, GBR: {GBR_MSE}")
print(f"R2:\n SVM: {SVM_r2}, GBR: {GBR_r2}")
```

```
MSE:
 SVM: 0.5871201495156598, GBR: 4.2681317856191895e-10
R2:
 SVM: 0.023202945570663158, GBR: 0.9999999992899071
```

Я использую две метрики - среднеквадратичную ошибку **(MSE)** и коэффициент детерминации **(R-squared).**

MSE измеряет среднеквадратичную разницу между фактическими и предсказанными значениями. Она предоставляет информацию о точности модели, где меньшее значение **MSE** указывает на лучшую модель. **MSE** полезна, когда мы хотим получить численную оценку ошибки модели.

R-squared измеряет долю объясненной дисперсии в данных. Он предоставляет информацию о том, насколько хорошо модель объясняет вариацию целевой переменной. Значение находится в диапазоне от **0** до **1**, чем ближе к **1**, тем лучше модель.

Вывод:

Модель градиентного бустинга очевидно лучше, так как **MSE** у нее намного меньше, а **R2** практически равна **1.**