Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по РК2

«*Методы построения моделей машинного обучения*»

по дисциплине «Технологии машинного обучения»

Выполнила:
студентка группы № ИУ5-62Б Кичикова Александра Олеговна

Проверил:
к.т.н., доц., Ю. Е. Гапанюк

2023 г.

Описание задачи. Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д. **Вариант № 10**

В работе был использован датасет brazilian_houses_to_rent. Файл содержит следующие колонки:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
```

```
df = pd.read_csv('houses_to_rent_v2.csv')
df
```

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	fire insurance (R$)	total (R$)
0	São Paulo	70	2	1	1	7	acept	furnished	2065	3300	211	42	5618
1	São Paulo	320	4	4	0	20	acept	not	1200	4960	1750	63	7973

Пропусков нет

```
df.isna().sum()
```

```
city                  0
area                  0
rooms                 0
bathroom              0
parking spaces        0
floor                 0
animal                0
furniture             0
hoa (R$)              0
rent amount (R$)      0
property tax (R$)     0
fire insurance (R$)   0
total (R$)            0
dtype: int64
```

Подготовка датасета к обучению модели. Колонки "hoa (R$)", "rent amount (R$)", "property tax (R$)", "fire insurance (R$)", “floor” были удалены, так как первые позже складываются в колонке, которую необходимо предсказать, а в последнем часто не указано значение

```
df = df.drop(["hoa (R$)", "rent amount (R$)", "property tax (R$)", "fire insurance (R$)", 'floor'], axis = 1)
df
```

	city	area	rooms	bathroom	parking spaces	animal	furniture	total (R$)
0	4	70	2	1	1	1	1	5618

Далее кодируем категориальные признаки.

```
# Кодируем категориальные признаки
le = LabelEncoder()
df['city'] = le.fit_transform(df['city'])
df['animal'] = df['animal'].replace('acept', 1)
df['animal'] = df['animal'].replace('not acept', 0)
df['furniture'] = df['furniture'].replace('furnished', 1)
df['furniture'] = df['furniture'].replace('not furnished', 0)
df
```

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa (R$)	rent amount (R$)	property tax (R$)	fire insurance (R$)	total (R$)
0	4	70	2	1	1	7	1	1	2065	3300	211	42	5618
1	4	320	4	4	0	20	1	0	1200	4960	1750	63	7973

После кодирования категориальных признаков выполним масштабирование данных в числовых колонках.

```
from sklearn.preprocessing import MinMaxScaler
# Масштабируем числовые признаки

scale_cols = ['area', 'parking spaces', 'rooms', 'bathroom', 'total (R$)']
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(df[scale_cols])
for i in range(len(scale_cols)):
    df[scale_cols[i]] = sc1_data[:,i]
df
```

	city	area	rooms	bathroom	parking spaces	animal	furniture	total (R$)
0	4	0.001274	0.083333	0.000000	0.083333	1	1	0.004573
1	4	0.006670	0.250000	0.333333	0.000000	1	0	0.006676

Создадим матрицу признаков и целевой переменной

```
# Создание матрицы признаков и целевой переменной
X = df.drop(columns='total (R$)')
y = df['total (R$)']
```

Обучаем модель классификатора (SVR).

```
# Разделяем данные на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80)

# Обучаем модель
svm_model = SVR()
svm_model.fit(X_train, y_train)

# Получаем предсказания на тестовом наборе
y_pred = svm_model.predict(X_test)

# Оцениваем качество модели
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('SVR MSE:', mse)
print('SVR R2:', r2)
```

```
SVR MSE: 0.009880003057237794
SVR R2: -9.100355292557447
```

В качестве метрик качества будем использовать mean squared error и r2. MSE измеряет среднюю сумму квадратной разности между фактическим значением и

прогнозируемым значением для всех точек данных. R2 это доля дисперсии зависимой переменной, объясняемая рассматриваемой моделью зависимости, то есть объясняющими переменными.

Заметим, что значения метрик говорят о низкой точности модели. Возможно, SVR не является оптимальной моделью для поставленной задачи.

Обучаем регрессионную модель (RandomForestRegressor).

```
# Разделение данных на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80)

# Обучение модели случайного леса
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Получение предсказаний на тестовом наборе
y_pred = rf.predict(X_test)

# Оценка качества модели
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Random Forest MSE:', mse)
print('Random Forest R2:', r2)
```

```
Random Forest MSE: 7.218881172649316e-05
Random Forest R2: 0.8490070670098557
```

Значения метрик значительно лучше для модели случайного леса, что говорит о том, что она более оптимальна для поставленной задачи