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 лабораторной работе №4

«Линейные модели, SVM и деревья решений**»**

Выполнил:
студент группы ИУ5-62Б
Перова Анна

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
    - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
    - SVM;
    - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Описание датасета

Датасет `SHOP-SALES.xlsx` содержит информацию о продажах магазина.

Параметры покупки:

- *Date* - дата покупки,
- *Hour* - час совершения покупки,
- *Product* - тип купленного товара,
- *Gender* - пол,
- *Color* - цвет товара,
- *Size* - размер товара,
- *Price* - цена товара без учета НДС,
- *Vat* - НДС,
- *Total* - Цена товара с НДС.

Подключение библиотек для анализа данных

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler, StandardScaler
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor, export_graphviz, export_text
from sklearn.svm import SVR
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV
from IPython.display import Image
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn import tree
from IPython.core.display import HTML
```

Загрузка датасета из файла `shop-sales.csv`

```
data = pd.read_excel('SHOP-SALES.xlsx')
```

Кодирование категорий товаров и времени суток

```
: le = LabelEncoder()
  data['TIME OF DAY'] = le.fit_transform(data['TIME OF DAY'])
```

```
: data['GENDER'] = le.fit_transform(data[ 'GENDER'])
```

```
: data['PRODUCT'] = le.fit_transform(data['PRODUCT'])
```

```
: data['SALE CONSULTANT'] = le.fit_transform(data['SALE CONSULTANT'])
```

```
: data['DAY_OF_WEEK'] = le.fit_transform(data['DAY_OF_WEEK'])
```

```
: data['CATEGORY'] = le.fit_transform(data['CATEGORY'])
```

```
: data['TIME OF DAY'].unique()
```

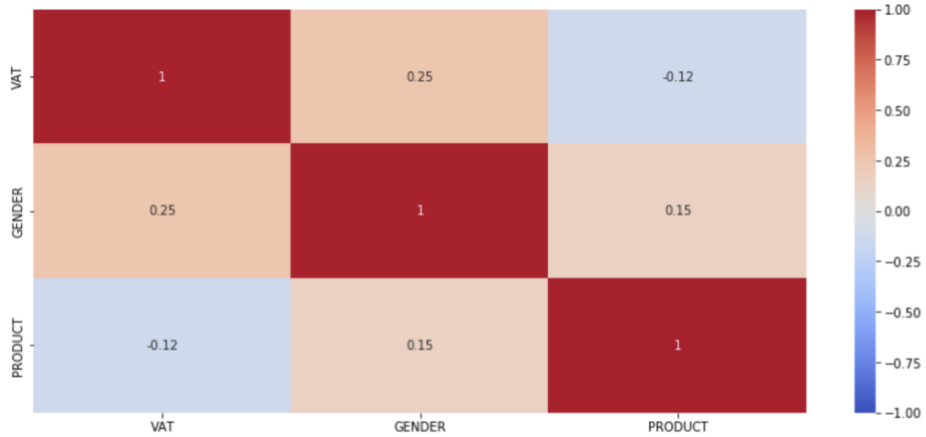
```
: array([2, 1, 0, 3])
```

```
print('Признаки, имеющие максимальную по модулю корреляцию с ценой товара')
best_params = data.corr()['PRICE'].map(abs).sort_values(ascending=False)[2:]
best_params = best_params[best_params.values >=0.15]
best_params
```

```
Признаки, имеющие максимальную по модулю корреляцию с ценой товара

VAT        0.856014
GENDER     0.297276
PRODUCT    0.180018
Name: PRICE, dtype: float64
```

```
plt.figure(figsize=(14, 6))
sns.heatmap(data[best_params.index].corr(), vmin=-1, vmax=1, cmap='coolwarm', annot=True)
plt.show()
```




```
plt.figure(figsize=(6, 3))
sns.heatmap(pd.DataFrame(data[np.append(best_params.index.values, 'PRICE')].corr()['PRICE'].sort_values(ascending=False
plt.show()
```

Разделим выборку на обучающую и тестовую

```
y = data['PRICE']
X = data[best_params.index]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

Линейная регрессия

```
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"MSE: {mean_squared_error(y_test, y_pred)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

```
linear_model = LinearRegression()
linear_model.fit(x_train, y_train)
y_pred_linear = linear_model.predict(x_test)
print_metrics(y_test, y_pred_linear)
```

```
R^2: 0.7548943636333737
MSE: 133.39130600899432
MAE: 6.475812350224802
```

SVM

```
scaler = StandardScaler().fit(x_train)
x_train_scaled = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test_scaled = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
x_train_scaled.describe()
```

	VAT	GENDER	PRODUCT
count	6.987000e+03	6.987000e+03	6.987000e+03
mean	-2.371519e-16	1.558475e-16	-2.128206e-16
std	1.000072e+00	1.000072e+00	1.000072e+00
min	-1.982651e+00	-1.915346e+00	-2.258018e+00
25%	-5.408781e-01	1.428703e-01	-8.658086e-01
50%	-1.804349e-01	1.428703e-01	1.467076e-01
75%	5.404514e-01	1.428703e-01	9.060948e-01
max	8.085403e+00	1.171979e+00	1.665482e+00

```
params = {'C': np.concatenate([np.arange(0.1, 2, 0.1), np.arange(2, 15, 1)])}
svm_model = SVR(kernel='linear')
grid_cv = GridSearchCV(estimator=svm_model, param_grid=params, cv=10, n_jobs=-1, scoring='r2')
grid_cv.fit(x_train_scaled, y_train)
print(grid_cv.best_params_)
```

```
{'C': 0.1}
```

```
best_svm_model = grid_cv.best_estimator_
best_svm_model = SVR(kernel='linear', C=0.1)
best_svm_model.fit(x_train_scaled, y_train)
y_pred_svm = best_svm_model.predict(x_test_scaled)
print_metrics(y_test, y_pred_svm)
```

```
R^2: 0.672134238577236
MSE: 178.43099310208575
MAE: 2.7740034015297934
```

Дерево решений

```
params = {'min_samples_leaf': range(3, 30)}
tree = DecisionTreeRegressor(random_state=3)
grid_cv = GridSearchCV(estimator=tree, cv=5, param_grid=params, n_jobs=-1, scoring='neg_mean_absolute_error')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'min_samples_leaf': 3}
```

```
best_tree = grid_cv.best_estimator_
best_tree.fit(x_train, y_train)
y_pred_tree = best_tree.predict(x_test)
print_metrics(y_test, y_pred_tree)
```

```
R^2: 0.9557031726131472
MSE: 24.10720432535853
MAE: 0.8030970551298966
```

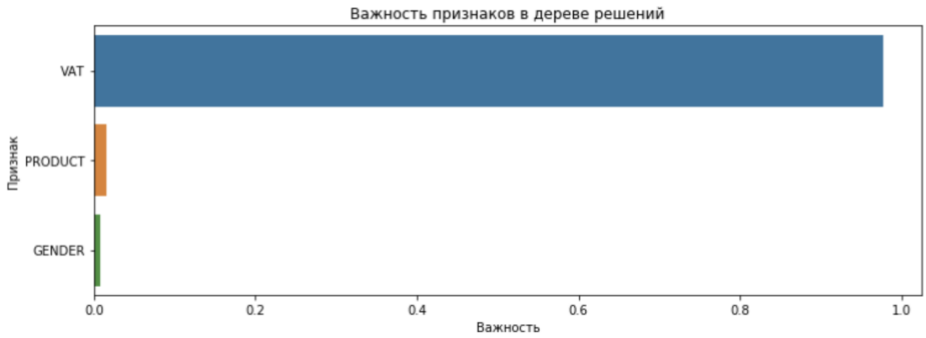
```
importances = pd.DataFrame(data=list(zip(x_train.columns, best_tree.feature_importances_)), columns=['Признак', 'Важнос
print('Важность признаков в дереве решений\n')
for row in importances.sort_values(by='Важность', ascending=False).values:
    print(f'{row[0]}: {round(row[1], 3)}')
```

```
Важность признаков в дереве решений

VAT: 0.977
PRODUCT: 0.015
GENDER: 0.008
```

```
plt.figure(figsize=(12, 4))
sns.barplot(data=importances.sort_values(by='Важность', ascending=False), y='Признак', x='Важность', orient='h', )
plt.title('Важность признаков в дереве решений')
plt.show()
```

```
plt.show()
```




```
# 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()
```

```
export_graphviz(best_tree, feature_names=best_params.index, filled=True, out_file='tree.dot')
```

```
!dot -Tpng tree.dot -o tree.png
Image(filename='tree.png')
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.892712 to fit
```

Сравнение моделей

```
print('Линейная регрессия')
print_metrics(y_test, y_pred_linear)

print('\nМетод опорных векторов')
print_metrics(y_test, y_pred_svm)

print('\nДерево решений')
print_metrics(y_test, y_pred_tree)
```

```
Линейная регрессия
R^2: 0.7548943636333737
MSE: 133.39130600899432
MAE: 6.475812350224802

Метод опорных векторов
R^2: 0.672134238577236
MSE: 178.43099310208575
MAE: 2.7740034015297934

Дерево решений
R^2: 0.9557031726131472
MSE: 24.10720432535853
MAE: 0.8030970551298966
```