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 ИУ5-61Б
Головацкий А.Д.

Проверил:
Гапанюк Ю. Е.

Москва, 2022 г.

ИУ5-61Б Головацкий Андрей

Линейные модели, SVM и деревья решений.

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.

```
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, DecisionTreeClassifier
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 IPython.core.display import HTML
```

```
data = pd.read_csv('C:\\Users\\Andrew\\Anaconda Projects\\datasets\\
computer_sales.csv')
# Удалим дубликаты записей, если они присутствуют
data = data.drop_duplicates()
# Удалим столбец-идентификатор
data = data.drop(columns=['laptop_ID'], axis=1)
```

```
# Также нас мало интересуют столбцы Memory2
data = data.loc[:, 'Company':'Memory1_type']
data.head()
```

```
  Company      Product   TypeName  Inches  Ram_GB  OpSys  Weight_kg  \
0   Apple  MacBook Pro  Ultrabook    13.3       8  macOS       1.37
1   Apple  Macbook Air  Ultrabook    13.3       8  macOS       1.34
2      HP       250 G6   Notebook    15.6       8  No OS       1.86
3   Apple  MacBook Pro  Ultrabook    15.4      16  macOS       1.83
4   Apple  MacBook Pro  Ultrabook    13.3       8  macOS       1.37

   Price_euros                ScreenType  ScreenWidth  ScreenHeight  \
0      1339.69  IPS Panel Retina Display         2560          1600
1       898.94                         -         1440           900
2       575.00                   Full HD         1920          1080
3      2537.45  IPS Panel Retina Display         2880          1800
4      1803.60  IPS Panel Retina Display         2560          1600

   ScreenRes             Cpu_type  Cpu_GHz Gpu_producer  \
0  2560x1600        Intel Core i5      2.3        Intel
1   1440x900        Intel Core i5      1.8        Intel
2  1920x1080  Intel Core i5 7200U      2.5        Intel
3  2880x1800        Intel Core i7      2.7          AMD
4  2560x1600        Intel Core i5      3.1        Intel

               Gpu_model  Memory1_GB   Memory1_type
0  Iris Plus Graphics 640         128            SSD
1        HD Graphics 6000         128  Flash Storage
2         HD Graphics 620         256            SSD
3          Radeon Pro 455         512            SSD
4  Iris Plus Graphics 650         256            SSD
```

```
# Список колонок с типами данных
data.dtypes
```

```
Company          object
Product          object
TypeName         object
Inches          float64
Ram_GB            int64
OpSys            object
Weight_kg       float64
Price_euros     float64
ScreenType       object
ScreenWidth       int64
ScreenHeight      int64
ScreenRes        object
Cpu_type         object
Cpu_GHz         float64
Gpu_producer     object
```

```
Gpu_model        object
Memory1_GB        int64
Memory1_type     object
dtype: object
```

```
# Проверим наличие пустых значений
data.isnull().sum()
```

```
Company         0
Product         0
TypeName        0
Inches          0
Ram_GB          0
OpSys           0
Weight_kg       0
Price_euros     0
ScreenType      0
ScreenWidth     0
ScreenHeight    0
ScreenRes       0
Cpu_type        0
Cpu_GHz         0
Gpu_producer    0
Gpu_model       0
Memory1_GB      0
Memory1_type    0
dtype: int64
```

Кодирование категориальных признаков

```
category_cols = ['Memory1_type', 'Company', 'Product', 'TypeName',
'OpSys',
                 'ScreenType', 'Cpu_type', 'Gpu_producer',
'Gpu_model', 'ScreenRes']

print('Количество уникальных значений')
for col in category_cols:
    print(f'{col}: {data[col].unique().size}')
```

```
Количество уникальных значений
Memory1_type: 4
Company: 19
Product: 618
TypeName: 6
OpSys: 9
ScreenType: 21
Cpu_type: 93
Gpu_producer: 4
Gpu_model: 110
ScreenRes: 15
```

Удалим столбцы, содержащие множество уникальных значений

```
remove_cols = ['Product', 'Gpu_model', 'Cpu_type']
for col in remove_cols:
    category_cols.remove(col)
data = pd.get_dummies(data, columns=category_cols)

data.drop(remove_cols, axis=1, inplace=True)
data.describe()
```

```
            Inches       Ram_GB    Weight_kg  Price_euros  ScreenWidth
\
count  1250.000000  1250.000000  1250.000000  1250.000000  1250.000000

mean     15.034880     8.443200     2.046152  1132.177480  1897.272000

std       1.416838     5.121929     0.669436   703.965444   491.854703

min      10.100000     2.000000     0.690000   174.000000  1366.000000

25%      14.000000     4.000000     1.500000   600.425000  1600.000000

50%      15.600000     8.000000     2.040000   985.000000  1920.000000

75%      15.600000     8.000000     2.310000  1489.747500  1920.000000

max      18.400000    64.000000     4.700000  6099.000000  3840.000000


       ScreenHeight      Cpu_GHz   Memory1_GB  Memory1_type_Flash
Storage  \
count   1250.000000  1250.000000  1250.000000
1250.000000
mean    1072.256000     2.303856   447.180800
0.055200
std      283.172078     0.502772   367.670259
0.228462
min      768.000000     0.900000     8.000000
0.000000
25%      900.000000     2.000000   256.000000
0.000000
50%     1080.000000     2.500000   256.000000
0.000000
75%     1080.000000     2.700000   512.000000
0.000000
max     2160.000000     3.600000  2048.000000
1.000000

       Memory1_type_HDD  ...  ScreenRes_2160x1440  ScreenRes_2256x1504
\
```

```
count         1250.0000  ...          1250.000000          1250.000000
mean             0.2848  ...             0.001600             0.004800
std              0.4515  ...             0.039984             0.069143
min              0.0000  ...             0.000000             0.000000
25%              0.0000  ...             0.000000             0.000000
50%              0.0000  ...             0.000000             0.000000
75%              1.0000  ...             0.000000             0.000000
max              1.0000  ...             1.000000             1.000000

       ScreenRes_2304x1440  ScreenRes_2400x1600
ScreenRes_2560x1440  \
count          1250.000000          1250.000000          1250.000000
mean              0.004800             0.003200             0.018400
std               0.069143             0.056501             0.134447
min               0.000000             0.000000             0.000000
25%               0.000000             0.000000             0.000000
50%               0.000000             0.000000             0.000000
75%               0.000000             0.000000             0.000000
max               1.000000             1.000000             1.000000

       ScreenRes_2560x1600  ScreenRes_2736x1824
ScreenRes_2880x1800  \
count          1250.000000          1250.000000          1250.000000
mean              0.004800             0.000800             0.003200
std               0.069143             0.028284             0.056501
min               0.000000             0.000000             0.000000
25%               0.000000             0.000000             0.000000
```

```
50%                0.000000             0.000000             0.000000

75%                0.000000             0.000000             0.000000

max                1.000000             1.000000             1.000000


       ScreenRes_3200x1800  ScreenRes_3840x2160
count          1250.000000          1250.000000
mean              0.020000             0.032800
std               0.140056             0.178184
min               0.000000             0.000000
25%               0.000000             0.000000
50%               0.000000             0.000000
75%               0.000000             0.000000
max               1.000000             1.000000

[8 rows x 86 columns]

data.head()

   Inches  Ram_GB  Weight_kg  Price_euros  ScreenWidth  ScreenHeight
Cpu_GHz  \
0    13.3       8       1.37      1339.69         2560          1600
2.3
1    13.3       8       1.34       898.94         1440           900
1.8
2    15.6       8       1.86       575.00         1920          1080
2.5
3    15.4      16       1.83      2537.45         2880          1800
2.7
4    13.3       8       1.37      1803.60         2560          1600
3.1

   Memory1_GB  Memory1_type_Flash Storage  Memory1_type_HDD  ...  \
0         128                           0                 0  ...
1         128                           1                 0  ...
2         256                           0                 0  ...
3         512                           0                 0  ...
4         256                           0                 0  ...

   ScreenRes_2160x1440  ScreenRes_2256x1504  ScreenRes_2304x1440  \
0                    0                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    0

   ScreenRes_2400x1600  ScreenRes_2560x1440  ScreenRes_2560x1600  \
0                    0                    0                    1
```

```
1                    0                    0                    0
2                    0                    0                    0
3                    0                    0                    0
4                    0                    0                    1

   ScreenRes_2736x1824  ScreenRes_2880x1800  ScreenRes_3200x1800  \
0                    0                    0                    0
1                    0                    0                    0
2                    0                    0                    0
3                    0                    1                    0
4                    0                    0                    0

   ScreenRes_3840x2160
0                    0
1                    0
2                    0
3                    0
4                    0

[5 rows x 86 columns]
```

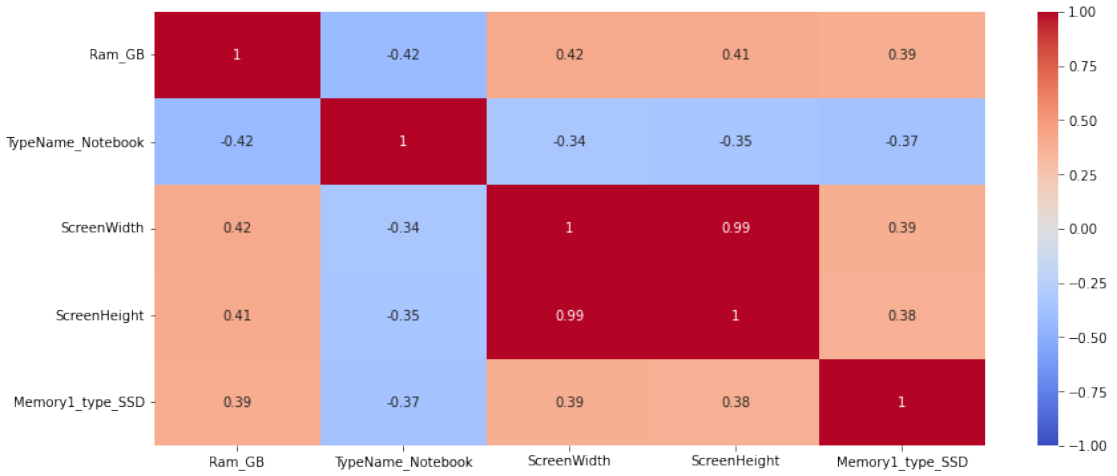
Корреляционный анализ

```
print('Признаки, имеющие максимальную по модулю корреляцию с ценой
компьютера')
best_params = data.corr()
['Price_euros'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.5]
best_params
```

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

Ram_GB               0.743141
TypeName_Notebook    0.555495
ScreenWidth          0.553660
ScreenHeight         0.550213
Memory1_type_SSD     0.505318
Name: Price_euros, 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_euros')].corr()['Price_euros'].sort_values(ascending=False)
[1:]), vmin=-1, vmax=1, cmap='coolwarm', annot=True)
plt.show()
```

Разделение выборки на обучающую и тестовую

```
y = data['Price_euros']
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

Напишем функцию для вывода основных метрик модели

```
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: 0.6519249821677147
MSE: 150718.0612966824
MAE: 283.2195714494765
```

Полиномиальная регрессия

```
poly_model = PolynomialFeatures(degree=3)
x_train_poly = poly_model.fit_transform(x_train)
x_test_poly = poly_model.fit_transform(x_test)
linear_model = LinearRegression()
linear_model.fit(x_train_poly, y_train)
y_pred_poly = linear_model.predict(x_test_poly)
print_metrics(y_test, y_pred_poly)
```

```
R^2: -48.53838660027276
MSE: 21450346.062342793
MAE: 749.58941766707
```

SVM

Проведем масштабирование выборок

```
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()
```

```
             Ram_GB  TypeName_Notebook   ScreenWidth  ScreenHeight  \
count  8.750000e+02       8.750000e+02  8.750000e+02  8.750000e+02
mean   1.015061e-16       1.248525e-16  1.827110e-16 -2.598556e-16
std    1.000572e+00       1.000572e+00  1.000572e+00  1.000572e+00
min   -1.213107e+00      -1.136035e+00 -1.057818e+00 -1.053280e+00
25%   -8.342815e-01      -1.136035e+00 -5.922109e-01 -5.957006e-01
50%   -7.663095e-02       8.802544e-01  4.451633e-02  2.827086e-02
75%   -7.663095e-02       8.802544e-01  4.451633e-02  2.827086e-02
max    1.053048e+01       8.802544e-01  3.864880e+00  3.772100e+00

       Memory1_type_SSD
count      8.750000e+02
mean       9.744586e-17
std        1.000572e+00
min       -1.339973e+00
25%       -1.339973e+00
50%        7.462838e-01
75%        7.462838e-01
max        7.462838e-01
```

```
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': 14.0}
```

```
best_svm_model = grid_cv.best_estimator_
best_svm_model = SVR(kernel='linear', C=11)
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: 0.6480040624498115
MSE: 152415.83731652604
MAE: 281.4248737876011
```

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: 0.6675552240622278
MSE: 143950.09567073712
MAE: 269.7408435691091
```

```
importances = pd.DataFrame(data=zip(x_train.columns,
best_tree.feature_importances_), columns=['Признак', 'Важность'])
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

Ram_GB: 0.823
TypeName_Notebook: 0.087
ScreenHeight: 0.032
```

```
ScreenWidth: 0.031
Memory1_type_SSD: 0.027
```

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


```
from io import StringIO
from IPython.display import Image
import graphviz
import pydotplus
from sklearn.tree import export_graphviz
```

```
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()
```

Вывод правил дерева решений

```
from IPython.core.display import HTML
from sklearn.tree import export_text
tree_rules = export_text(best_tree,
feature_names=list(best_params.index))
HTML('<pre>' + tree_rules + '</pre>')
```

```
<IPython.core.display.HTML object>
```

Сравнение построенных моделей

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

print('\nПолиномиальная регрессия')
print_metrics(y_test, y_pred_poly)
```

```
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: 0.6519249821677147
MSE: 150718.0612966824
MAE: 283.2195714494765

Полиномиальная регрессия
R^2: -48.53838660027276
MSE: 21450346.062342793
MAE: 749.58941766707

Метод опорных векторов
R^2: 0.6480040624498115
MSE: 152415.83731652604
MAE: 281.4248737876011

Дерево решений
R^2: 0.6675552240622278
MSE: 143950.09567073712
MAE: 269.7408435691091
```