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 вычислительная техника»
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

Веревкина Диана В.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Цель лабораторной работы

Изучение линейных моделей, SVM и деревьев решений.

Описание задания

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.

Текст программы и результаты ее выполнения

Импорт библиотек и базы данных

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
import warnings
warnings.simplefilter("ignore", FutureWarning)

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 IPython.core.display import HTML


df = pd.read_csv('penguins_size.csv')
df.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

Обработка пропусков

```
#обработка пропусков
#поиск пропусков
df.isna()
df.dropna()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
...	...	...	...	...	...	...	...
338	Gentoo	Biscoe	47.2	13.7	214.0	4925.0	FEMALE
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

334 rows × 7 columns

Общая информация о базе данных

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   culmen_length_mm   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

Кодирование категориальных признаков

```
#Кодирование категориальных признаков
df["species"].value_counts()
df["species"] = df["species"].astype('category')
df["island"] = df["island"].astype('category')
df["sex"] = df["sex"].astype('category')

#Назначим закодированную переменную новому столбцу с помощью метода доступа cat.codes
df["species_cat"] = df["species"].cat.codes + 1
df["island_cat"] = df["island"].cat.codes +1
df["sex_cat"] = df["sex"].cat.codes
df.isna()
df.dropna()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex	species_cat	island_cat	sex_cat
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE	1	3	2
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE	1	3	1
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE	1	3	1
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE	1	3	1
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE	1	3	2
...	...	...	...	...	...	...	...	...	...	...
338	Gentoo	Biscoe	47.2	13.7	214.0	4925.0	FEMALE	3	1	1
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE	3	1	1
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE	3	1	2
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE	3	1	1

```
df_cat=df.drop(['species', 'island', 'sex'], axis=1, inplace=True)
df.dropna()
```

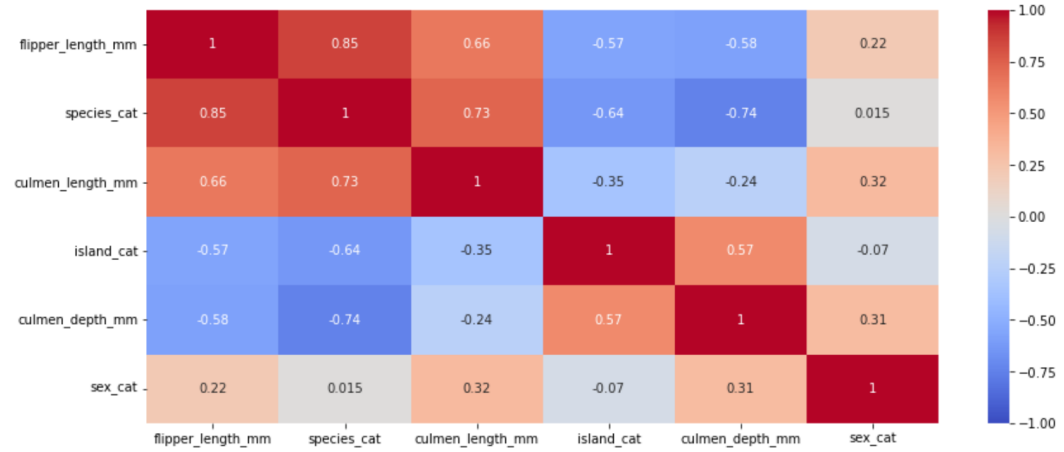
	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_cat	island_cat	sex_cat
0	39.1	18.7	181.0	3750.0	1	3	2
1	39.5	17.4	186.0	3800.0	1	3	1
2	40.3	18.0	195.0	3250.0	1	3	1
4	36.7	19.3	193.0	3450.0	1	3	1
5	39.3	20.6	190.0	3650.0	1	3	2
...	...	...	...	...	...	...	...
338	47.2	13.7	214.0	4925.0	3	1	1
340	46.8	14.3	215.0	4850.0	3	1	1
341	50.4	15.7	222.0	5750.0	3	1	2
342	45.2	14.8	212.0	5200.0	3	1	1
343	49.9	16.1	213.0	5400.0	3	1	2

Корреляционный анализ

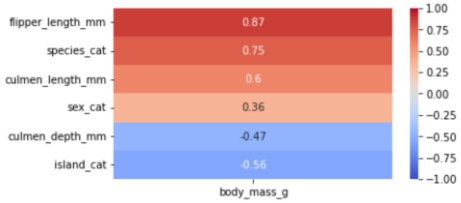
```
print('Признаки, имеющие максимальную по модулю корреляцию с массой пингвина')
best_params = df.corr()['body_mass_g'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.35]
best_params
```

```
Признаки, имеющие максимальную по модулю корреляцию с массой пингвина
flipper_length_mm    0.871202
species_cat          0.750491
culmen_length_mm     0.595110
island_cat           0.561515
culmen_depth_mm      0.471916
sex_cat              0.361138
Name: body_mass_g, dtype: float64
```

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




```
plt.figure(figsize=(6, 3))
sns.heatmap(pd.DataFrame(df[np.append(best_params.index.values, 'body_mass_g')].corr()['body_mass_g'].sort_values(ascending=False)[1:]), vmin=-1, vmax=1, cmap='coolwarm
plt.show()
```




```
df.isna()
df = df.dropna()
df
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_cat	island_cat	sex_cat
0	39.1	18.7	181.0	3750.0	1	3	2
1	39.5	17.4	186.0	3800.0	1	3	1
2	40.3	18.0	195.0	3250.0	1	3	1
4	36.7	19.3	193.0	3450.0	1	3	1
5	39.3	20.6	190.0	3650.0	1	3	2
...	...	...	...	...	...	...	...
338	47.2	13.7	214.0	4925.0	3	1	1
340	46.8	14.3	215.0	4850.0	3	1	1
341	50.4	15.7	222.0	5750.0	3	1	2
342	45.2	14.8	212.0	5200.0	3	1	1
343	49.9	16.1	213.0	5400.0	3	1	2

342 rows × 7 columns

Разделение выборки на обучающую и тестовую

```
#Разделение выборки на обучающую и тестовую
y=df['body_mass_g'] #.to_numpy()
#X=df.drop('body_mass_g', axis=1)
X = df[best_params.index] #.to_numpy()
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

```
type(x_train)
```

```
pandas.core.frame.DataFrame
```

Линейная регрессия

```
#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: 0.8053336022242322
MSE: 133355.3359533476
MAE: 286.30311274644066
```

Пополиномиальная регрессия

```
#Пополиномиальная 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: -9.542844467415676
MSE: 7222327.951409407
MAE: 563.880529579607
```

SVM

```
#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()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	species_cat	island_cat	sex_cat
count	2.380000e+02	2.380000e+02	2.380000e+02	2.400000e+02	2.400000e+02	2.400000e+02
mean	3.246703e-16	1.177629e-15	8.447958e-16	-1.295260e-17	3.747003e-17	-3.700743e-17
std	1.002107e+00	1.002107e+00	1.002107e+00	1.002090e+00	1.002090e+00	1.002090e+00
min	-1.893278e+00	-2.056978e+00	-2.051675e+00	-1.027118e+00	-9.159459e-01	-3.597007e+00
25%	-8.626114e-01	-7.332903e-01	-7.938938e-01	-1.027118e+00	-9.159459e-01	-6.201737e-01
50%	2.340006e-02	8.128656e-02	-2.348798e-01	8.327980e-02	4.753643e-01	-6.201737e-01
75%	8.325636e-01	7.431303e-01	8.831481e-01	1.193677e+00	4.753643e-01	8.682431e-01
max	2.826089e+00	2.219551e+00	2.071053e+00	1.193677e+00	1.866675e+00	8.682431e-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: 0.8052968350375075
MSE: 133380.52314843496
MAE: 295.17537702718096
```

Дерево решений

```
#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': 7}
```

```
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)
```

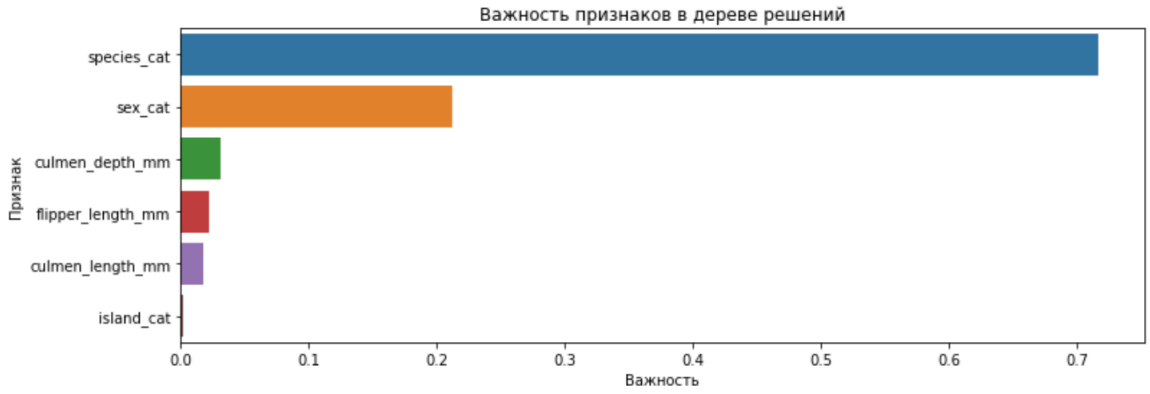
```
R^2: 0.8287563226807713
MSE: 117309.70716937499
MAE: 273.12736414071367
```

```
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

species_cat: 0.716
sex_cat: 0.212
culmen_depth_mm: 0.031
flipper_length_mm: 0.022
culmen_length_mm: 0.018
island_cat: 0.001
```

```
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()
```



Сравнение моделей

```
#Сравнение 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: 0.8053336022242322
MSE: 133355.3359533476
MAE: 286.30311274644066

Полиномиальная регрессия
R^2: -9.542844467415676
MSE: 7222327.951409407
MAE: 563.880529579607

Метод опорных векторов
R^2: 0.8052968350375075
MSE: 133380.52314843496
MAE: 295.17537702718096

Дерево решений
R^2: 0.8287563226807713
MSE: 117309.70716937499
MAE: 273.12736414071367
```

Вывод

В ходе выполнения данной лабораторной работы я повторила язык программирования Python и работу с юпитер тетрадками. Также изучила линейные модели, SVM и деревья решений.