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 № 5
«Ансамбли моделей машинного обучения»

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 ансамблей моделей машинного обучения.

Описание задания

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
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 ансамблевые модели:
    - одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
    - одну из моделей группы бустинга;
    - одну из моделей группы стекинга.
5. **(+1 балл на экзамене)** Дополнительно к указанным моделям обучите еще две модели:
    - Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.
    - Модель МГУА с использованием библиотеки - https://github.com/kvoyager/GmdhPy (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей.
6. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы и результаты ее выполнения

Импорт библиотек и базы данных

```
from sklearn.preprocessing import PolynomialFeatures, MinMaxScaler, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset
from sklearn.neural_network import MLPRegressor
from gmdhpy import gmdh
from warnings import simplefilter
```

```
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)

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

	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

Общая информация о базе данных

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 334 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            334 non-null    object 
 1   island             334 non-null    object 
 2   culmen_length_mm   334 non-null    float64
 3   culmen_depth_mm    334 non-null    float64
 4   flipper_length_mm  334 non-null    float64
 5   body_mass_g        334 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 20.9+ 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
```

	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
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE	3	1	2

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

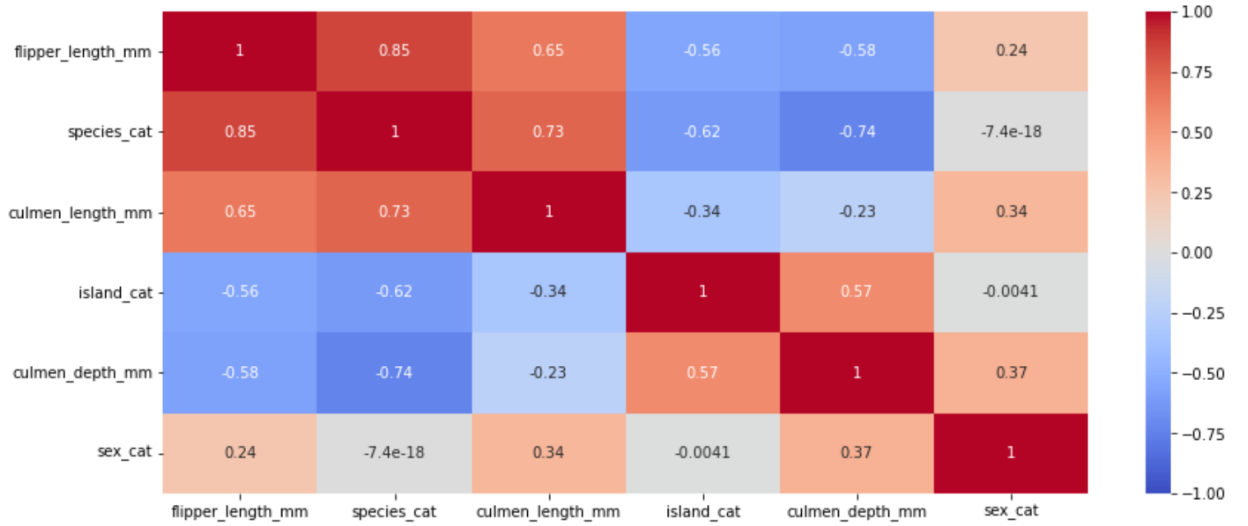
	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_cat	island_cat	sex_cat
0	39.1	18.7	181.0	3750.0	1	3	2
1	39.5	17.4	186.0	3800.0	1	3	1
2	40.3	18.0	195.0	3250.0	1	3	1
4	36.7	19.3	193.0	3450.0	1	3	1
5	39.3	20.6	190.0	3650.0	1	3	2
...	...	...	...	...	...	...	...
338	47.2	13.7	214.0	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

Корреляционный анализ

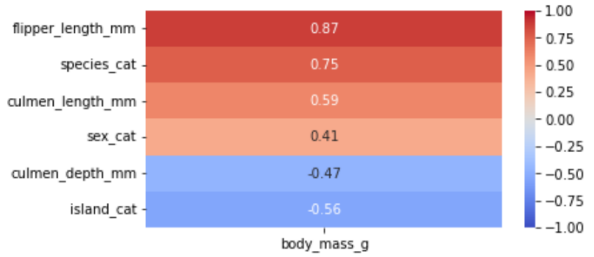
```
print('Признаки, имеющие максимальную по модулю корреляцию с массой пингвина')
best_params = df.corr()['body_mass_g'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.35]
best_params
```

```
Признаки, имеющие максимальную по модулю корреляцию с массой пингвина
flipper_length_mm    0.873211
species_cat          0.751020
culmen_length_mm     0.589066
island_cat           0.560518
culmen_depth_mm      0.472987
sex_cat              0.411531
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)[
plt.show()
```



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)
```

Случайный лес

```
#Случайный лес
print_metrics(y_test, RandomForestRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.8579528606883411
MSE: 92408.04517326732
MAE: 245.52722772277227
```

```
#Подбор гиперпараметров
rf = RandomForestRegressor(random_state=17)
params = {'n_estimators': [100, 1000], 'criterion': ['squared_error', 'absolute_error', 'poisson'],
          'max_features': ['auto', 'sqrt'], 'min_samples_leaf': [1, 3, 5]}
grid_cv = GridSearchCV(estimator=rf, 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_)
```

```
{'criterion': 'poisson', 'max_features': 'sqrt', 'min_samples_leaf': 3, 'n_estimators': 1000}
```

```
best_rf = grid_cv.best_estimator_
best_rf.fit(x_train, y_train)
y_pred_rf = best_rf.predict(x_test)
print_metrics(y_test, y_pred_rf)
```

```
R^2: 0.8510894074822187
MSE: 96873.029804351
MAE: 242.719270568073
```

Градиентный бустинг

```
#Градиентный бустинг
print_metrics(y_test, GradientBoostingRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: 0.8581324685280105
MSE: 92291.20220520701
MAE: 249.18006668994684
```

```
#Подбор гиперпараметров
gb = GradientBoostingRegressor(random_state=17)
params = {'loss': ['squared_error', 'absolute_error', 'huber'], 'n_estimators': [10, 50, 100, 200],
          'criterion': ['friedman_mse', 'squared_error', 'mse', 'mae'], 'min_samples_leaf': [1, 3, 5]}
grid_cv = GridSearchCV(estimator=gb, cv=5, param_grid=params, n_jobs=-1, scoring='r2')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'criterion': 'mae', 'loss': 'huber', 'min_samples_leaf': 5, 'n_estimators': 200}
```

```
best_gb = grid_cv.best_estimator_
best_gb.fit(x_train, y_train)
y_pred_gb = best_gb.predict(x_test)
print_metrics(y_test, y_pred_gb)
```

```
R^2: 0.8726256011050022
MSE: 82862.76840240907
MAE: 228.54916987291918
```

Стекинг

```
#Стекинг
dataset = Dataset(x_train, y_train, x_test)

model_lr = Regressor(dataset=dataset, estimator=LinearRegression, name='lr')
model_rf = Regressor(dataset=dataset, estimator=RandomForestRegressor,
                     parameters={'criterion': 'absolute_error', 'n_estimators': 1000, 'random_state': 17}, name='rf')
model_gb = Regressor(dataset=dataset, estimator=GradientBoostingRegressor,
                     parameters={'loss': 'huber', 'random_state': 17}, name='rf')
```

```
pipeline = ModelsPipeline(model_lr, model_rf)
stack_ds = pipeline.stack(k=10, seed=1)
stacker = Regressor(dataset=stack_ds, estimator=GradientBoostingRegressor)
results = stacker.validate(k=10, scorer=mean_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [191.39424868214826, 223.5971668487191, 216.86376824238184, 272.97770520828004, 275.15017993431206, 225.94857678271197, 236.46696841469
94, 268.2067712261299, 198.22714693578052, 262.2396119721167]
Mean accuracy: 237.10721442472794
Standard Deviation: 29.413190351082335
Variance: 865.1357666290029
```

```
y_pred_stack = stacker.predict()
print_metrics(y_test, y_pred_stack)
```

```
R^2: 0.7207185369761542
MSE: 120930.14007496767
MAE: 247.18161038788267
```

Многослойный персептрон

```
#Многослойный персептрон
print_metrics(y_test, MLPRegressor(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
R^2: -26.41465952246979
MSE: 17834467.542523753
MAE: 4147.488503121823
```

```
#Подбор гиперпараметров
mlp = MLPRegressor(random_state=17)
params = {'solver': ['lbfgs', 'sgd', 'adam'], 'hidden_layer_sizes': [(100,), (50, 30,), (100, 40,)],
          'alpha': [1e-4, 3e-4, 5e-4], 'max_iter': [500, 1000]}
grid_cv = GridSearchCV(estimator=mlp, cv=5, param_grid=params, n_jobs=-1, scoring='r2')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'alpha': 0.0003, 'hidden_layer_sizes': (100, 40), 'max_iter': 500, 'solver': 'lbfgs'}
```

```
best_mlp = grid_cv.best_estimator_
best_mlp.fit(x_train, y_train)
y_pred_mlp = best_mlp.predict(x_test)
print_metrics(y_test, y_pred_mlp)
```

```
R^2: 0.8697886925455147
MSE: 84708.3048601505
MAE: 229.34859973675194
```

Метод группового учёта аргументов

```
gm = gmdh.Regressor(n_jobs=-1)
gm.fit(np.array(x_train_scaled), np.array(y_train))
y_pred_gm = gm.predict(np.array(x_test_scaled))
print()
print_metrics(y_test, y_pred_gm)
```

```
train layer0 in 0.01 sec
train layer1 in 0.05 sec
train layer2 in 0.04 sec
train layer3 in 0.05 sec
train layer4 in 0.04 sec
train layer5 in 0.05 sec
train layer6 in 0.04 sec
train layer7 in 0.04 sec
train layer8 in 0.03 sec

R^2: 0.6642449299187112
MSE: 145383.4680475877
MAE: 274.30940411915725
```

Сравнение моделей

```
#Сравнение моделей
print("Случайный лес")
print_metrics(y_test, y_pred_rf)

print("\nГрадиентный бустинг")
print_metrics(y_test, y_pred_gb)

print("\nСтекинг")
print_metrics(y_test, y_pred_stack)

print("\nМногослойный персептрон")
print_metrics(y_test, y_pred_mlp)

print("\nМетод группового учёта аргументов")
print_metrics(y_test, y_pred_gm)
```

```
Случайный лес
R^2: 0.8510894074822187
MSE: 96873.029804351
MAE: 242.719270568073

Градиентный бустинг
R^2: 0.8726256011050022
MSE: 82862.76840240907
MAE: 228.54916987291918
Стекинг
R^2: 0.7207185369761542
MSE: 120930.14007496767
MAE: 247.18161038788267

Многослойный персептрон
R^2: 0.6422646017371612
MSE: 154901.0498344665
MAE: 288.659695272951

Метод группового учёта аргументов
R^2: 0.6642449299187112
MSE: 145383.4680475877
MAE: 274.30940411915725
```

Вывод

В ходе выполнения данной лабораторной работы я повторила язык программирования Python и работу с юпитер тетрадками. Также изучила ансамбли моделей машинного обучения.