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 и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

ЛР №5

По курсу
«Технологии машинного обучения»

Подготовил:

Студент группы

ИУ5-63Б Борисов А.М.

08.04.2022

Проверил:

2022 г.

Задание

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.

Решение:

Лабораторная работа №5: Ансамбли моделей машинного обучения.

```
[ ] #Датасет содержит данные о кредитах на покупку электроники, которые были одобрены Tinkoff.ru.
    import pandas as pd
    import numpy as np
    from matplotlib import pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.preprocessing import MinMaxScaler, StandardScaler
    from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
    from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, mean_absolute_error
    from heamy.estimator import Classifier
    from heamy.pipeline import ModelsPipeline
    from heamy.dataset import Dataset
    from sklearn.neural_network import MLPClassifier
    from gmdhpy import gmdh
    from warnings import simplefilter

    simplefilter('ignore')
```

```
[ ] !pip install heamy
    !pip install gmdhpy
```

```
[ ] # записываем CSV-файл в объект DataFrame
    data = pd.read_csv('credit_train_preprocess.csv', encoding='cp1251', sep=',')
```

```
[ ] # смотрим на первые пять строк
    data.head()
```

	age	credit_sum	credit_month	tariff_id	score_shk	monthly_income	credit_count	overdue_credit_count	open_account_
0	34.0	59998.00	10	1.6	0.461599	30000.0	1.0	1.0	
1	34.0	10889.00	6	1.1	0.461599	35000.0	2.0	0.0	
2	32.0	10728.00	12	1.1	0.461599	35000.0	5.0	0.0	
3	27.0	12009.09	12	1.1	0.461599	35000.0	2.0	0.0	
4	45.0	21229.00	10	1.1	0.421385	35000.0	1.0	0.0	

5 rows × 39 columns

Корреляционный анализ

```
[ ] corr = data.corr().round(2)
    f, ax = plt.subplots(figsize=(20, 20))
    cmap = sns.diverging_palette(220, 10, as_cmap=True)
    sns.heatmap(data=corr, cmap=cmap, annot=True, vmax=1.0, square=True, linewidths=.3, cbar_kws={"shrink": .5}
    plt.show()
```


```
[ ] print('Признаки, имеющие максимальную по модулю корреляцию с целевым признаком')
    best_params = data.corr()['open_account_flg'].map(abs).sort_values(ascending=False)[1:]
    best_params = best_params[best_params.values > 0.02]
    best_params
```

```
Признаки, имеющие максимальную по модулю корреляцию с целевым признаком
education_GRD          0.084832
education_SCH          0.079952
job_position_PNA       0.077167
credit_sum             0.071824
tariff_id              0.071803
marital_status_MAR     0.070222
marital_status_UNM     0.062804
score_shk              0.051117
job_position_SPC       0.048190
gender_F               0.042854
gender_M               0.042854
job_position_ATP       0.036842
age                    0.034142
credit_month           0.028047
credit_count           0.027126
monthly_income         0.024567
job_position_NOR       0.022414
job_position_BIU       0.020819
Name: open_account_flg, dtype: float64
```

Разделение выборки на обучающую и тестовую

```
[ ] data_best = data[best_params.index]
    data_best.head()
```

	education_GRD	education_SCH	job_position_PNA	credit_sum	tariff_id	marital_status_MAR	marital_status_UNM	score_shk	job_position_SPC	gender_F	gend
0	1.0	0.0	0.0	59998.00	1.6	1.0	0.0	0.461599	0.0	0.0	
1	0.0	1.0	0.0	10889.00	1.1	1.0	0.0	0.461599	0.0	1.0	
2	0.0	1.0	0.0	10728.00	1.1	1.0	0.0	0.461599	1.0	0.0	
3	0.0	1.0	0.0	12009.09	1.1	1.0	0.0	0.461599	1.0	1.0	
4	0.0	1.0	0.0	21229.00	1.1	1.0	0.0	0.421385	1.0	0.0	

```
[ ] y = data['open_account_flg']
    #X = data.drop('open_account_flg', axis=1)
    X = data_best
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.75, random_state=21)
    x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.3, random_state=21)
```

Масштабирование данных

```
[ ] scaler = MinMaxScaler().fit(x_train)
    x_train = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
    x_test = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
    x_train.describe()
```

Модель №1: Случайный лес

```
[ ] def print_metrics(y_test, y_pred):
        print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
        print(f"Precision: {precision_score(y_test, y_pred)}")
        print(f"Recall: {recall_score(y_test, y_pred)}")
        print(f"F1-measure: {f1_score(y_test, y_pred)}")
```

```
[ ] print_metrics(y_test, RandomForestClassifier(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
Accuracy: 0.8274052478134111
Precision: 0.47619047619047616
Recall: 0.09090909090909091
F1-measure: 0.15267175572519084
```

Подбор гиперпараметров

```
[ ] rf = RandomForestClassifier(random_state=17)
    params = {'n_estimators': [100, 1000],
              '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_)
```

```
{'max_features': 'auto', 'min_samples_leaf': 5, 'n_estimators': 100}
```

```
[ ] 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)
```

```
Accuracy: 0.8316812439261418
Precision: 0.6060606060606061
Recall: 0.045454545454545456
F1-measure: 0.08456659619450317
```

Модель №2: Градиентный бустинг

```
[ ] print_metrics(y_test, GradientBoostingClassifier(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
Accuracy: 0.8218803685772295
Precision: 0.5819672131147541
Recall: 0.06118052563550194
F1-measure: 0.11072124756335285
```

Подбор гиперпараметров

```
[ ] gb = GradientBoostingClassifier(random_state=17)
    params = {'n_estimators': [10, 50, 100, 200], 'min_samples_leaf': [1, 3, 5]}
    grid_cv = GridSearchCV(estimator=gb, cv=5, param_grid=params, n_jobs=-1, scoring='f1')
    grid_cv.fit(x_train, y_train)
    print(grid_cv.best_params_)
```

```
{'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)
```

```
Accuracy: 0.8301263362487852
Precision: 0.5194805194805194
Recall: 0.09090909090909091
F1-measure: 0.15473887814313347
```

Модель №3: Стекинг

```
[ ] dataset = Dataset(x_train, y_train, x_test)
```

```
[ ] model_lr = Classifier(dataset=dataset, estimator=LogisticRegression, name='lr')
    model_rf = Classifier(dataset=dataset, estimator=RandomForestClassifier,
                          parameters={'n_estimators': 1000, 'random_state': 17}, name='rf')
    model_gb = Classifier(dataset=dataset, estimator=GradientBoostingClassifier,
                          parameters={'random_state': 17}, name='gb')
```

```
    pipeline = ModelsPipeline(model_lr, model_gb, model_rf)
    stack_ds = pipeline.stack(k=10, seed=1)
    stacker = Classifier(dataset=stack_ds, estimator=RandomForestClassifier)
```

```
[ ] results = stacker.validate(k=10, scorer=mean_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [0.25095046854083, 0.2522289156626506, 0.2640428380187416, 0.2544444444444444, 0.2506994645247657, 0.2504484605087015
Mean accuracy: 0.25231091030789826
Standard Deviation: 0.004632756433221238
Variance: 2.1462432169552774e-05
```

```
[ ] p_pred = stacker.predict()
    y_pred_stack = np.where(p_pred > 0.5, 1, 0)
    print_metrics(y_test, y_pred_stack)
```

```
Accuracy: 0.8194363459669582
Precision: 0.41694915254237286
Recall: 0.13977272727272727
F1-measure: 0.20936170212765956
```

Модель №4: Многослойный персептрон

```
[ ] print_metrics(y_test, MLPClassifier(random_state=17).fit(x_train, y_train).predict(x_test))
```

```
Accuracy: 0.8197719818834921
Precision: 0.5241635687732342
Recall: 0.06074967686342094
F1-measure: 0.10888030888030888
```

Подбор гиперпараметров

```
[ ] mlp = MLPClassifier(random_state=17)
    params = {'solver': ['lbfgs', 'sgd', 'adam'], 'hidden_layer_sizes': [(15,), (30, 15,), (40, 20,)],
              'alpha': [1e-4, 3e-4, 5e-4], 'max_iter': [500]} #, 'max_iter': [500, 1000]
    grid_cv = GridSearchCV(estimator=mlp, cv=5, param_grid=params, n_jobs=-1, scoring='f1')
    grid_cv.fit(x_train, y_train)
    print(grid_cv.best_params_)
```

```
{'alpha': 0.0005, 'hidden_layer_sizes': (40, 20), '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)
```

```
Accuracy: 0.814382896015549
Precision: 0.3665480427046263
Recall: 0.11704545454545455
F1-measure: 0.17743324720068904
```

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)
```

```
Случайный лес
Accuracy: 0.8316812439261418
Precision: 0.6060606060606061
Recall: 0.045454545454545456
F1-measure: 0.08456659619450317

Градиентный бустинг
Accuracy: 0.8301263362487852
Precision: 0.5194805194805194
Recall: 0.09090909090909091
F1-measure: 0.15473887814313347

Стекинг
Accuracy: 0.8194363459669582
Precision: 0.41694915254237286
Recall: 0.13977272727272727
F1-measure: 0.20936170212765956

Многослойный персептрон
Accuracy: 0.814382896015549
Precision: 0.3665480427046263
Recall: 0.11704545454545455
F1-measure: 0.17743324720068904
```

Сравнение моделей: наилучшие показатели - Случайный лес