МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра прикладной математики (ПМ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Технологии и инструментарий машинного обучения»

Практическая работа № 3

Студент группы ИМБО-01-21 Малкина В.В. _____
(подпись)

Старший преподаватель Высоцкая А.А. _____
(подпись)

Отчет представлен «___»_____2023г.

Москва 2023 г.

1 Предобработка данных

Выведем общую информацию о столбцах из набора, также выведем уникальные значения для каждого столба (Рисунок 1).

```
#найти и скачать многомерные данные для классификации
data = pd.read_csv('/Users/vladamalkina/Downloads/Employee.csv')
```

```
#предобработать, проверить выбросы, пропуски, есть ли дисбаланс классов и прочее
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4653 entries, 0 to 4652
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Education                  4653 non-null   object
 1   JoiningYear                4653 non-null   int64 
 2   City                       4653 non-null   object
 3   PaymentTier                4653 non-null   int64 
 4   Age                        4653 non-null   int64 
 5   Gender                     4653 non-null   object
 6   EverBenched                4653 non-null   object
 7   ExperienceInCurrentDomain  4653 non-null   int64 
 8   LeaveOrNot                 4653 non-null   int64 
dtypes: int64(5), object(4)
memory usage: 327.3+ KB
```

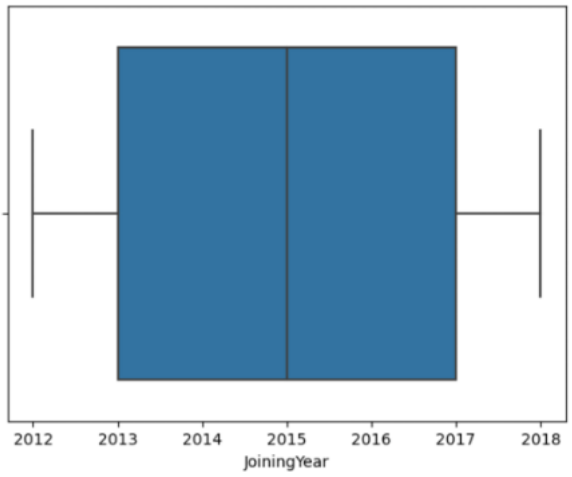
```
for col in data.columns:
    print(data[col].unique())
```

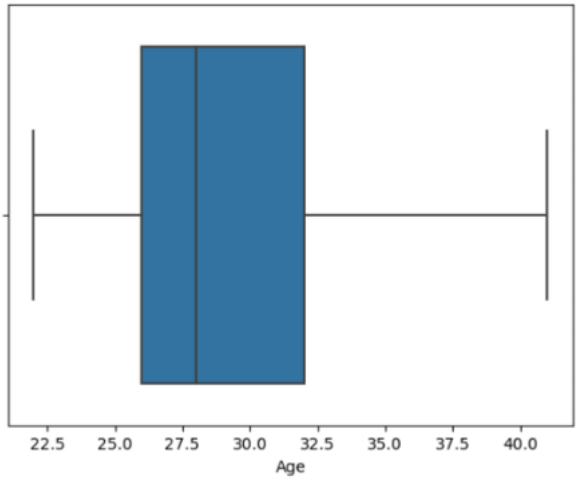
```
['Bachelors' 'Masters' 'PHD']
[2017 2013 2014 2016 2015 2012 2018]
['Bangalore' 'Pune' 'New Delhi']
[3 1 2]
[34 28 38 27 24 22 23 37 32 39 29 30 36 31 25 26 40 35 33 41]
['Male' 'Female']
['No' 'Yes']
[0 3 2 5 1 4 7 6]
[0 1]
```

Рисунок 1 — Вывод общей информации

Исследуем данные на выбросы, построим график «ящик с усами» (Рисунок 2).

```
col = ['JoiningYear', 'Age', 'ExperienceInCurrentDomain']
for c in col:
    sns.boxplot(x=data[c])
    plt.show()
```





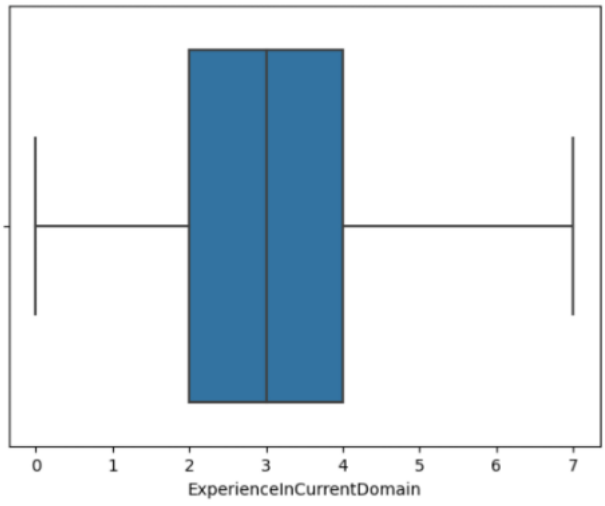


Рисунок 2 — Графики «Ящики с усами»

Проверим классы на дисбаланс (Рисунок 3).

```
class_frequency = data_train['LeaveOrNot'].value_counts(normalize = True)
print(class_frequency)
class_frequency.plot(kind='bar')
```

```
0    0.653143
1    0.346857
Name: LeaveOrNot, dtype: float64

<AxesSubplot:>
```

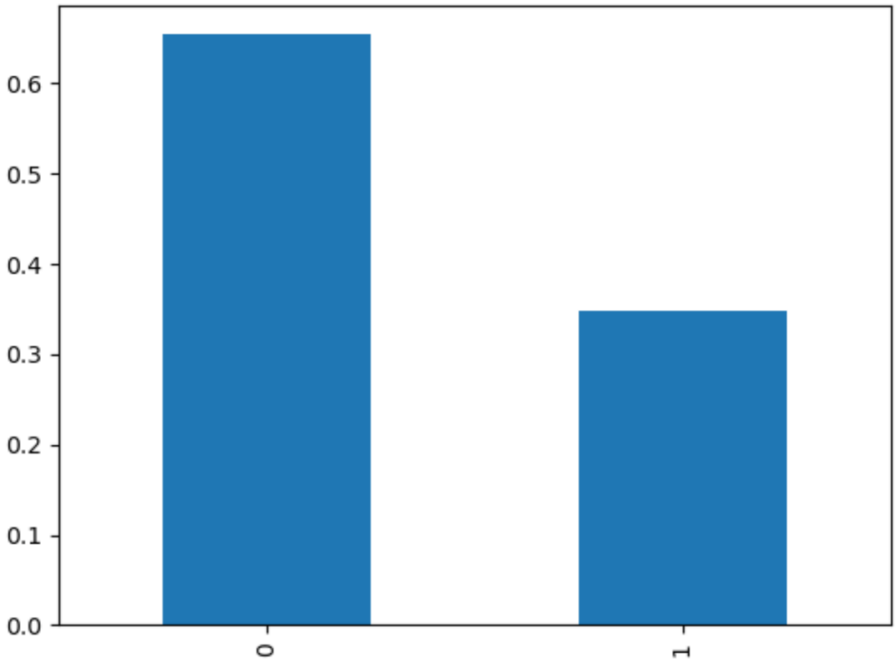


Рисунок 3 — Дисбаланс классов

Разделим данные на обучающую и тестовую выборку (Рисунок 4).

```
x_train = data_train.drop('LeaveOrNot', axis = 1)
y_train = data_train['LeaveOrNot']
x_test = data_test.drop('LeaveOrNot', axis = 1)
y_test = data_test['LeaveOrNot']
```

```
categorical_cols = data.select_dtypes(include=["object"]).columns.tolist()
print(len(categorical_cols))
categorical_cols
```

```
4

['Education', 'City', 'Gender', 'EverBenched']
```

```
numeric_cols = data.select_dtypes(exclude=["object"]).columns.tolist()
print(len(numeric_cols))
numeric_cols
target_name = 'LeaveOrNot'
numeric_cols.remove(target_name)
```

```
5
```

Рисунок 4 —Разделение данных на выборки

Закодируем категориальные признаки и нормализуем численные (Рисунок 5).

```
#сделаем одну выборку с нормализованными численными признаками
from sklearn.preprocessing import StandardScaler

X_train_num = x_train[numeric_cols]
X_test_num = x_test[numeric_cols]

scaler = StandardScaler()
X_train_num_scaled = scaler.fit_transform(X_train_num)
X_test_num_scaled = scaler.transform(X_test_num)
```

```
# Кодирование категориальных признаков
from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder(handle_unknown='use_encoded_value', unknown_value=-1)

X_train_cat = encoder.fit_transform(x_train[categorical_cols]).astype(int)
X_test_cat = encoder.fit_transform(x_test[categorical_cols]).astype(int)
```

```
x_train = np.hstack((X_train_num_scaled, X_train_cat))
x_test = np.hstack((X_test_num_scaled, X_test_cat))
x_train.shape, x_test.shape
```

```
((3722, 8), (931, 8))
```

Рисунок 5 — Кодирование признаков

2 Обучение моделей.

Построим логистическую регрессию (Рисунок 6)

Логистическая регрессия

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression() #class_weight = 'balanced'
model.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_pred = model.predict(x_test)
```

Рисунок 6 — Обучение логистической регрессии

Теперь выведем метрики и матрицу ошибок (Рисунок 6).

```python
#вывести метрики accuracy, f1-score, PR и ROC кривые, PR и ROC AUC, матрицу ошибок
#может пригодиться from sklearn.metrics import classification_report

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))


from sklearn.metrics import roc_auc_score
probabilities = model.predict_proba(x_test)
proba = probabilities[:, 1]
print("AUC-ROC модели:", roc_auc_score(y_test, proba))
print()

print(sklearn.metrics.confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.85      0.79       622
           1       0.57      0.40      0.47       309

    accuracy                           0.70       931
   macro avg       0.65      0.62      0.63       931
weighted avg       0.68      0.70      0.68       931

AUC-ROC модели: 0.6935399952132697

[[528  94]
 [186 123]]
```

Рисунок 6 — Метрики, матрица ошибок

Построим PR и ROC кривые (Рисунок 7-8).

```python
from sklearn.preprocessing import label_binarize
from sklearn.metrics import precision_recall_curve, roc_curve, auc
y_bin = label_binarize(y_test, classes=[0, 1])
precision, recall, _ = precision_recall_curve(y_bin.ravel(), proba.ravel())
false_positive_rate, true_positive_rate, _ = roc_curve(y_bin.ravel(), proba.ravel())
pr_auc = auc(recall, precision)
roc_auc = auc(false_positive_rate, true_positive_rate)
```

```python
# Plotting PR Curve
plt.figure()
plt.plot(recall, precision, color='purple', label=f'PR Curve (AUC = {pr_auc:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='lower right')
plt.show()

# Plotting ROC Curve
plt.figure()
plt.plot(false_positive_rate, true_positive_rate, color='red', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

Рисунок 7 — Построение PR и ROC кривых

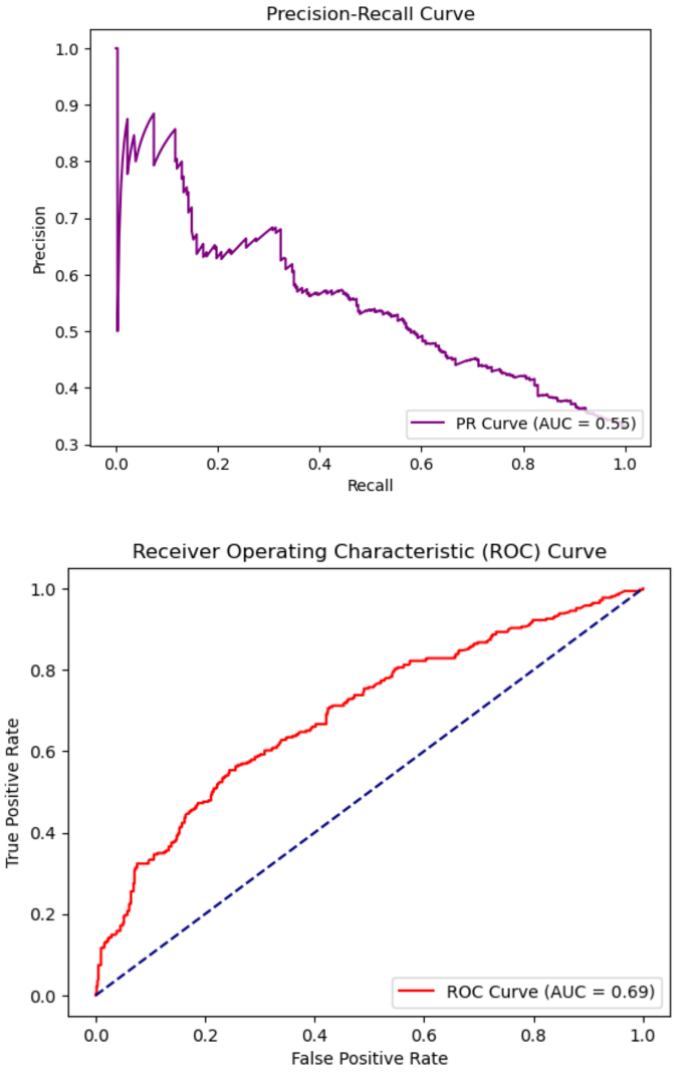


Рисунок 8 — PR и ROC кривые

Теперь натренируем модель KNN на предобработаных данных (Рисунок 9).

KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

```
# использовать алгоритм KNN из sklearn
```

```
#количество соседей для перебора
number_of_neighbors = np.arange(1, 10, 1)

#инициализировать модель
knn = KNeighborsClassifier()

param_grid = {'n_neighbors': number_of_neighbors}

#использовать GridSearchCV для поиска оптимального гиперпараметра
grid_search = GridSearchCV(knn, param_grid, cv=5)
```

```
grid_search.fit(x_train, y_train) #обучение модели
```

```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([1, 2, 3, 4, 5, 6, 7, 8, 9])})
```

```
best_params = grid_search.best_params_
```

```
best_params
```

```
{'n_neighbors': 4}
```

```
grid_search.best_estimator_ # лучшая модель получается при k = 4
```

```
KNeighborsClassifier(n_neighbors=4)
```

```
best_model = KNeighborsClassifier(**best_params)
best_model.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=4)
```

Рисунок 9 — Обучение KNN модели

Выведем метрики и матрицу ошибок для данной модели (Рисунок 10).

```
pred = best_model.predict(x_test) # результат работы модели на тесте
```

```
#вывести метрики accuracy, f1-score, PR и ROC кривые, PR и ROC AUC, матрицу ошибок

print(classification_report(y_test, pred))

probabilities = best_model.predict_proba(x_test)
proba = probabilities[:, 1]
print("AUC-ROC модели:", roc_auc_score(y_test, proba))
print()
print(sklearn.metrics.confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.94      0.87       622
           1       0.82      0.58      0.68       309

    accuracy                           0.82       931
   macro avg       0.82      0.76      0.77       931
weighted avg       0.82      0.82      0.81       931

AUC-ROC модели: 0.8183305757604138

[[528  94]
 [186 123]]
```

Рисунок 10 — Метрики, матрица ошибок

Теперь построим PR и ROC кривые (Рисунок 11).

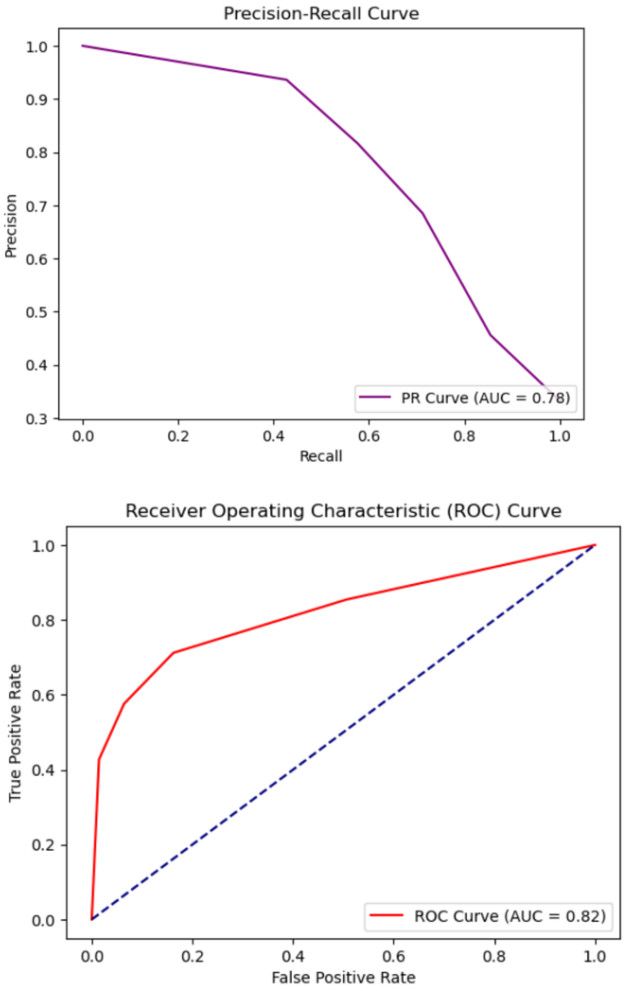


Рисунок 11 — PR и ROC кривые

Теперь на тех же данных обучим SVM модель (Рисунок 12).

SVM

```
from sklearn.svm import SVC
```

```
#использовать SVM для решения задачи классификации
param_kernel = ('linear', 'rbf', "poly", "sigmoid") # ядра

svm = SVC()

param_grid = {'kernel': param_kernel}

grid_search = GridSearchCV(estimator=svm, param_grid=param_grid, cv=5)
grid_search.fit(x_train, y_train)
```

```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'kernel': ('linear', 'rbf', 'poly', 'sigmoid')})
```

```
best_params = grid_search.best_params_
```

```
best_params
```

```
{'kernel': 'rbf'}
```

```
best_model = SVC(**best_params, probability=True)
best_model.fit(x_train, y_train)
```

```
SVC(probability=True)
```

```
y_pred = best_model.predict(x_test) # результат работы модели на тесте
```

Рисунок 12 — Обучение SVM модели

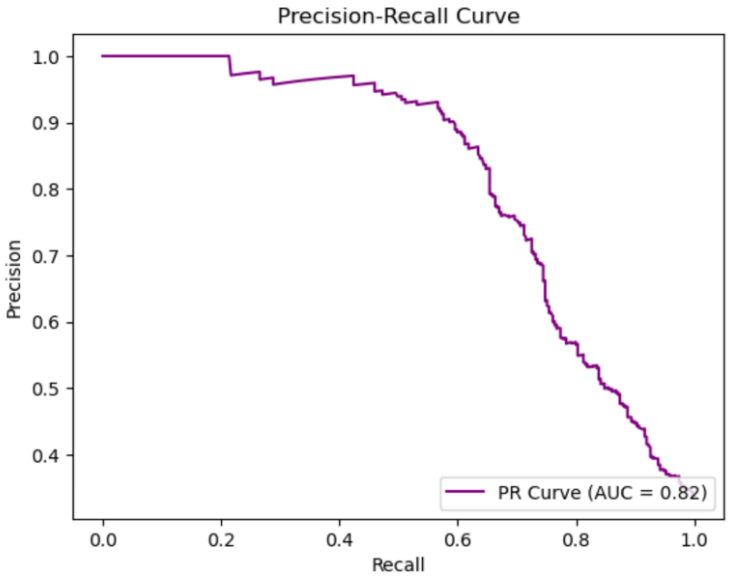
Выведем метрики и матрицу ошибок (Рисунок 13).

```
#вывести метрики accuracy, f1-score, PR и ROC кривые, PR и ROC AUC, матрицу ошибок
print(classification_report(y_test, y_pred))

print(sklearn.metrics.confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.95      0.89       622
           1       0.87      0.61      0.72       309

    accuracy                           0.84       931
   macro avg       0.85      0.78      0.80       931
weighted avg       0.84      0.84      0.83       931

[[593  29]
 [120 189]]
```

Рисунок 13 — Метрики, матрица ошибок

Построим PR и ROC кривые (Рисунок 14).



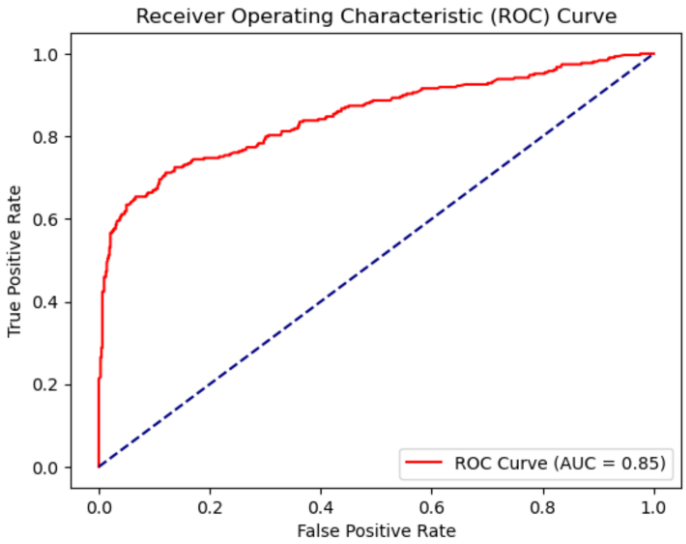


Рисунок 14 — PR и ROC кривые

Выводы

Модели	accuracy	f1-мера	pr-auc	roc-auc
Логистическая регрессия	0.7	0.47	0.55	0.69
Метод ближайших соседей	0.82	0.68	0.78	0.82
Метод опорных векторов	0.84	0.72	0.82	0.85

Мы научились решать задачу классификации используя различные методы. Лучшие результаты классификации получились у метода опорных векторов. Вторая по эффективности модель – метод ближайших соседей. Худшие результаты мы получили на логистической регрессии.