МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА СИСТЕМ СБОРА И ОБРАБОТКИ ДАННЫХ

НГТУ
НЭТИ

Лабораторная работа №2

по дисциплине «Машинное обучение»

на тему

«Метод случайного леса»

Группа: АТМ-25

Факультет: АВТФ

Студент: Секачёв Г. М.

Преподаватель: Павлова А. И.

Новосибирск, 2025

СОДЕРЖАНИЕ

1. Обработка данных

Файл с данными датасета https://archive.ics.uci.edu/dataset/2/adult не содержит заголовков, так что дадим их самостоятельно.

```
col_names = [ #указать ссылку где взяли датасет
 'age', 'workclass', 'fnlwgt', 'education', 'education-num',
 'marital-status', 'occupation', 'relationship', 'race', 'sex',
 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income'
]
```

Считаем данные файла **data.csv**, вставив заголовки.

```
df = pd.read_csv('data.csv',
                 sep=',', # разделитель данных в файле
                 #header=0, # номер строки с заголовками, нумерация с нуля
                 header=None, # если заголовки отсутствуют
                 names=col_names,
                 skipinitialspace=True,
                 na_values='?',
                 comment='|'
                )
print("Размер таблицы", df.shape)
df[:2]
```

```
Размер таблицы (32561, 15)
```

Столбцы содержат числовые и строковые данные, в некоторых столбцах присутствуют значения NaN, уберём их.

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       30725 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      30718 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours-per-week  32561 non-null  int64 
 13  native-country  31978 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
None
```

```
df.isna().sum()
```

```
age                0
workclass       1836
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation      1843
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   583
income             0
dtype: int64
```

```
df = df.dropna()
print("Размер таблицы:", df.shape)
df.isna().sum()
```

```
Размер таблицы: (30162, 15)

age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

2. Разделение данных на обучающую и тестовую выборки

Разделим данные на обучающую и тестовую выборки. Для этого выделим числовые и категориальные признаки и создадим конвейер (pipeline) предобработки данных. В нём для числовых признаков применим нормализацию с помощью StandardScaler, а для категориальных признаков – кодирование методом OneHotEncoder, позволяющим преобразовать категориальные значения в бинарные признаки. Перед разделением данных выделим матрицу признаков x, исключив из исходного набора столбец 'income', и сформируем целевую переменную y, преобразовав значения '<=50K' и '>50K' в бинарный формат (0 и 1 соответственно).Далее с помощью функции train_test_split разобьём данные на обучающую и тестовую части в соотношении 80/20, сохранив пропорции классов целевой переменной с помощью параметра stratify=y.

```
numeric_features = [
    'age', 'fnlwgt', 'education-num',
    'capital-gain', 'capital-loss', 'hours-per-week'
]

categorical_features = [
    'workclass', 'education', 'marital-status', 'occupation',
    'relationship', 'race', 'sex', 'native-country'
]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
# Трансформер для разных типов признаков
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('scaler', StandardScaler())
        ]), numeric_features),

        ('cat', Pipeline([
            ('onehot', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_features)
    ]
)
```

```
x = df.drop(columns=['income'])
y = df['income'].apply(lambda s: 1 if s.strip() == '>50K' else 0)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=0, stratify=y #соотношение
)
```

3. Выбор значимых признаков с помощью метода случайного леса

Для определения наиболее значимых признаков была использована модель случайного леса (Random Forest Classifier). Данный метод основан на ансамбле деревьев решений, где каждый классификатор строится на случайной подвыборке признаков и объектов. После обучения модели оценивается вклад каждого признака в процесс разделения узлов дерева с помощью показателя важности признаков (feature importance), который отражает, насколько данный признак снижает неопределённость в классификации.

В работе были обучены две модели случайного леса с различными критериями разбиения:

- **критерий Джини (Gini)** – оценивает чистоту узлов на основе вероятностей классов;
- **энтропия (Entropy)** – измеряет уровень неопределённости в данных.

После обучения для каждой модели были вычислены значения важности признаков (feature_importances_) и сформированы таблицы с ранжированными признаками по степени их вклада в классификацию.
На основании этих результатов были выделены наиболее значимые признаки, оказывающие наибольшее влияние на прогноз дохода (income). Для наглядности были выведены топ-20 признаков по важности для каждого критерия.

```
rf = RandomForestClassifier(
    n_estimators=100,
    criterion='gini',      # entropy
    random_state=0,
    bootstrap=True,
    max_features='sqrt',
    n_jobs=-1
)

pipe_gini = make_pipeline(preprocessor, rf)
pipe_gini.fit(x_train, y_train)

# извлекаем имена признаков после one-hot кодирования
ct = pipe_gini.named_steps['columntransformer']
num_names = numeric_features
ohe = ct.named_transformers_['cat'].named_steps['onehot']
ohe_names = list(ohe.get_feature_names_out(categorical_features))
feature_names = num_names + ohe_names

# извлекаем важности признаков
importances = pipe_gini.named_steps['randomforestclassifier'].feature_importances_
feat_imp_gini = pd.DataFrame({'feature': feature_names, 'importance': importances})
feat_imp_gini = feat_imp_gini.sort_values('importance', ascending=False).reset_index(drop=True)

# топ-20 признаков
print(feat_imp_gini.head(20))
```

Gini / entropy

	feature	importance
0	fnlwgt	0.159269
1	age	0.152573
2	capital-gain	0.096988
3	hours-per-week	0.082646
4	education-num	0.059496
5	marital-status_Married-civ-spouse	0.055940
6	relationship_Husband	0.046891
7	capital-loss	0.028727
8	marital-status_Never-married	0.023041
9	occupation_Exec-managerial	0.018574
10	occupation_Prof-specialty	0.013899
11	education_Bachelors	0.013281
12	relationship_Not-in-family	0.012617
13	sex_Female	0.011799
14	workclass_Private	0.010893
15	relationship_Wife	0.010793
16	sex_Male	0.010192
17	education_Masters	0.008954
18	workclass_Self-emp-not-inc	0.008801
19	relationship_Own-child	0.008275

	feature	importance
0	fnlwgt	0.160174
1	age	0.159222
2	capital-gain	0.090027
3	hours-per-week	0.084405
4	marital-status_Married-civ-spouse	0.054317
5	education-num	0.052336
6	relationship_Husband	0.040380
7	capital-loss	0.027345
8	marital-status_Never-married	0.027066
9	occupation_Exec-managerial	0.016775
10	relationship_Not-in-family	0.014275
11	occupation_Prof-specialty	0.013860
12	relationship_Wife	0.012979
13	relationship_Own-child	0.012883
14	sex_Female	0.012088
15	workclass_Private	0.011560
16	sex_Male	0.010676
17	education_Bachelors	0.010659
18	workclass_Self-emp-not-inc	0.008763
19	occupation_Other-service	0.008614

4. Выполнение задачи классификации методом случайного леса

Для решения задачи классификации была обучена модель Random Forest Classifier с гиперпараметрами по умолчанию (100 деревьев решений, критерий Джини, использование бутстрепа, random_state=0 для воспроизводимости).

После обучения модели на обучающей выборке был выполнен прогноз на тестовых данных.

```
y_pred = pipe_gini.predict(x_test)
y_proba = pipe_gini.predict_proba(x_test)[:, 1]  # вероятность класса 1 (>50K)
```

Полученные предсказания использовались для оценки качества модели с помощью нескольких метрик:

- **Матрица ошибок:** с помощью функции confusion_matrix() была построена матрица классификации, показывающая количество правильных и ошибочных прогнозов по каждому классу.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[4185  346]
 [ 568  934]]
```

  - **4185** – количество верно классифицированных примеров класса <=50K (True Negatives);
  - **934** – количество верно классифицированных примеров класса >50K (True Positives);
  - **346** – количество примеров с доходом <=50K, ошибочно отнесённых к >50K (False Positives);
  - **568** – количество примеров с доходом >50K, ошибочно классифицированных как <=50K (False Negatives).

Таким образом, модель **лучше определяет лиц с доходом ≤50K**, чем с доходом >50K, что объясняется несбалансированностью классов (преобладание низкого дохода в выборке).

- **Отчёт по классификации:** с помощью функции classification_report() были вычислены метрики Precision, Recall и F1-score для каждого класса (<=50K и >50K).

```python
from sklearn.metrics import classification_report, accuracy_score
# отчёт по классификации
creport = classification_report(y_test, y_pred, target_names=['<=50K','>50K'])
print("Classification Report:\n", creport)
```

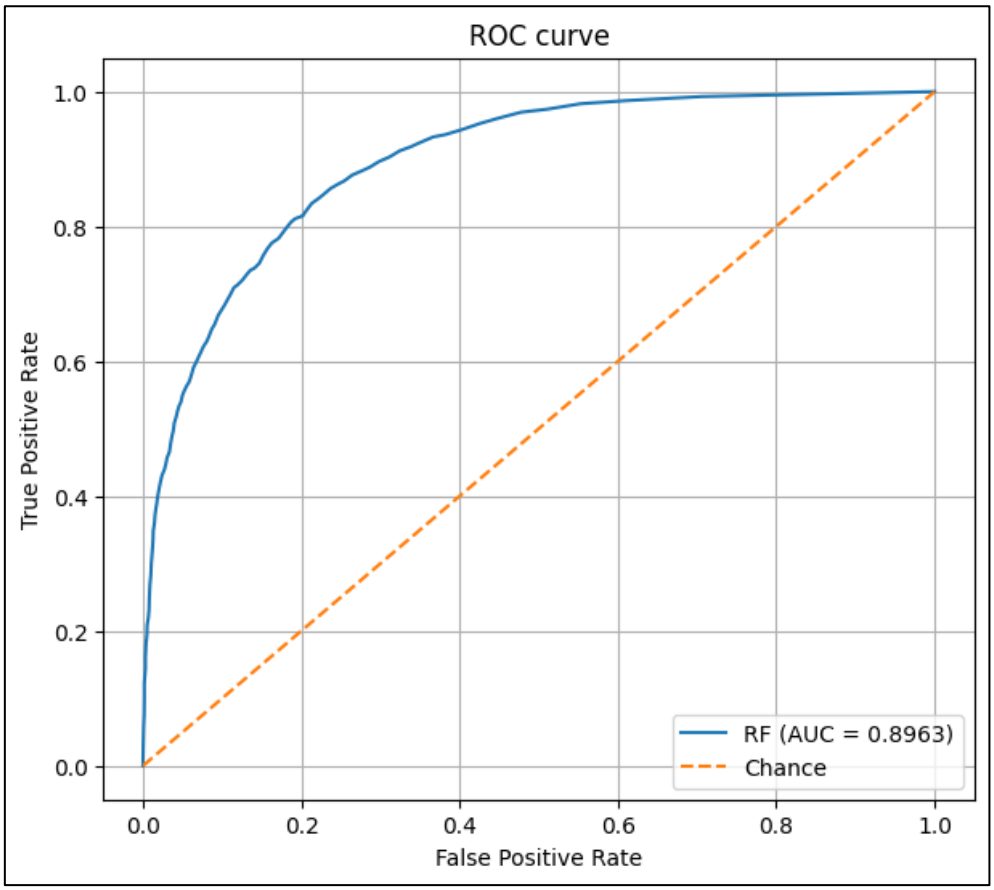
```
Classification Report:
              precision    recall  f1-score   support

       <=50K       0.88      0.92      0.90      4531
        >50K       0.73      0.62      0.67      1502

    accuracy                           0.85      6033
   macro avg       0.81      0.77      0.79      6033
weighted avg       0.84      0.85      0.84      6033
```

- **Precision (точность)** показывает долю правильно предсказанных объектов среди всех предсказанных как данный класс.
  Для >50K точность = 0.73 – значит, 73% всех предсказаний «высокого дохода» оказались верными.
- **Recall (полнота)** показывает, какую долю объектов данного класса модель смогла найти.
  Для >50K полнота = 0.62 – модель определила 62% всех людей с высоким доходом.
- **F1-score** — гармоническое среднее между precision и recall. Чем ближе значение F1-score к 1, тем лучше модель справляется с задачей классификации, учитывая обе метрики Precision и Recall. Если F1-score равен 0, это означает, что модель полностью не справляется с задачей классификации.
- **Accuracy** = 0.85 – общая доля верных классификаций, что говорит о достаточно высоком качестве модели.

- **ROC-кривая и AUC:** для анализа способности модели различать классы была построена ROC-кривая и вычислено значение AUC. Чем ближе AUC к 1, тем лучше модель различает классы.



На графике ROC-кривой видно, что кривая заметно отклоняется вверх от диагонали случайного угадывания («Chance»), а значение **AUC = 0.8963** указывает на **высокую способность модели различать два класса**. Это означает, что вероятность того, что случайно выбранный пример с доходом >50K будет классифицирован моделью выше, чем пример с доходом ≤50K, составляет примерно **89,6%**.

- **Общая точность классификации:** общая доля правильных прогнозов модели вычислялась с помощью accuracy_score().

```
# общая точность классификации
acc_100 = accuracy_score(y_test, y_pred)
print('Model accuracy score with 100 decision-trees : {0:0.4f}'.format(acc_100))
```

```
Model accuracy score with 100 decision-trees : 0.8485
```

Модель с 100 деревьями решений достигла **точности 84.85%**, что является хорошим результатом для задачи бинарной классификации на реальных социально-экономических данных (Adult dataset).

5. Создание набора данных и переобучение модели

Для анализа результатов классификации был сформирован датафрейм, содержащий следующие столбцы:

1. **y_true** – истинные значения целевой переменной из тестовой выборки,
2. **y_pred** – прогнозные значения, полученные с помощью обученной модели,
3. **abs_error** – абсолютная ошибка для каждой наблюдаемой записи, вычисленная как $| y_{true} - y_{pred} |$. Для задач классификации эта метрика показывает, где модель ошиблась.

```
# сохранение таблицы с y_true, y_pred и абсолютной ошибкой
results_df = x_test.copy().reset_index(drop=True)
results_df['y_true'] = y_test.reset_index(drop=True)
results_df['y_pred'] = y_pred
# для классификации абсолютная ошибка как |y_true - y_pred|
results_df['abs_error'] = (results_df['y_true'] - results_df['y_pred']).abs()

results_df.to_csv("adult_classification_results.csv", index=False)
print("Saved results to adult_classification_results.csv (содержит y_true, y_pred, abs_error)")
```

В рамках переобучения модели:

- Рассчитана важность признаков по критерию **Gini**.
- Выбраны **ТОП-10** наиболее значимых признаков.
- На основе выбранных топ-признаков была переобучена модель Random Forest, после чего вычислена точность на тестовой выборке.

```
TOP_K = 10
top_features = feat_imp_gini['feature'].head(TOP_K).tolist()
print("Top features by Gini (top {}):".format(TOP_K), top_features)

x_train_trans = pipe_gini.named_steps['columntransformer'].transform(x_train)
x_test_trans = pipe_gini.named_steps['columntransformer'].transform(x_test)

# Преобразуем в плотный формат
if hasattr(x_train_trans, "toarray"):
    x_train_trans = x_train_trans.toarray()
    x_test_trans = x_test_trans.toarray()

feature_names = list(feat_imp_gini['feature'])

# Формируем DataFrame из трансформированных признаков
x_train_trans_df = pd.DataFrame(x_train_trans, columns=feature_names)
x_test_trans_df = pd.DataFrame(x_test_trans, columns=feature_names)

# берём top-k
x_train_topk = x_train_trans_df[top_features]
x_test_topk = x_test_trans_df[top_features]

# переобучаем RF на топ-k признаков
rf_topk = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)
rf_topk.fit(x_train_topk, y_train)
y_pred_topk = rf_topk.predict(x_test_topk)
print("Accuracy on top-{} features: {:.4f}".format(TOP_K, accuracy_score(y_test, y_pred_topk)))
```

Результаты показали, что даже с использованием только 10 наиболее значимых признаков модель сохраняет высокую точность, что подтверждает информативность выбранных признаков.