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»

КАФЕДРА «Системы обработки информации и управления»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Решение задачи классификации»

Студент ИУ5-62Б _____ **Федюкин Д.А.**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

2020 г.

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)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ5
(Индекс)
_____ В.М. Черненький
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине «Технологии машинного обучения»

Студент группы ИУ5-62Б

Федюкин Данила Антонович
(Фамилия, имя, отчество)

Тема курсового проекта «Решение задачи классификации»

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 16 нед.

Задание Решение задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Оформление курсового проекта:

Расчетно-пояснительная записка на 25 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « 7 » февраля 2020 г.

Руководитель курсового проекта _____ **Ю.Е. Гапанюк**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **Федюкин Д.А.**
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Введение 4

Основная часть 5

Задание 5

Последовательность действий 5

1) Поиск и выбор набора данных для построения моделей машинного обучения 5

2) Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных 7

4) Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения 14

5) Выбор метрик для последующей оценки качества моделей 15

6) Выбор наиболее подходящих моделей для решения задачи классификации 16

7) Формирование обучающей и тестовой выборок на основе исходного набора данных…. 16

8) Построение базового решения (baseline) для выбранных моделей. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки 16

11) Формирование выводов о качестве построенных моделей на основе выбранных метрик 16

Заключение 24

Список использованных источников 25

Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовой проект опирается на знания, умения и владения, полученные в рамках лекций и лабораторных работ по дисциплине.

В рамках курсового проекта было проведено типовое исследование – решение задачи машинного обучения на основе материалов дисциплины.

Основная часть

Задание

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

1) Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.

2) Проведение разведочного анализа данных.

3) Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.

4) Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.

5) Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.

6) Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.

7) Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.Формирование обучающей и тестовой выборок на основе исходного набора данных.

8) Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

9) Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

10) Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

11) Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Приведенная схема исследования является рекомендуемой. В зависимости от решаемой задачи возможны модификации.

Последовательность действий

В этом проекте я пытаюсь делать прогнозы, где задача прогнозирования состоит в том, чтобы определить, зарабатывает ли человек более 50 тысяч долларов в год. Я реализую классификацию случайных лесов с помощью Python и Scikit-Learn.

Я использую датасет "**Income classification data set**".

```
[ ] !pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.6/dist-packages (0.23.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from catboost) (1.4.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from catboost) (3.2.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.6/dist-packages (from catboost) (1.18.4)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from catboost) (1.12.0)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.6/dist-packages (from catboost) (1.0.3)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (from catboost) (0.10.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from catboost) (4.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost) (2.8.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost) (0.10.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.24.0->catboost) (2018.9)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly->catboost) (1.3.3)
```

```
[ ] import seaborn as sns
    import matplotlib.pyplot as plt
    import pandas as pd
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import RobustScaler
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.linear_model import  LogisticRegression, LogisticRegressionCV
    from sklearn.svm import  SVC
    from catboost import CatBoostClassifier
    from sklearn.pipeline import Pipeline
    import category_encoders as ce
    from sklearn.metrics import accuracy_score, roc_auc_score, mean_absolute_error


    %matplotlib inline
```

```
[ ] sns.set(style="whitegrid")
```

```
[ ] df = pd.read_csv("/content/income_evaluation.csv")
```

Разведочный анализ данных

```
[ ] print('размер датасета : ', df.shape)
```

```
размер датасета :  (32561, 15)
```

```
[ ] df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

```
[ ] col_names = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship',
                 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'income']
    df.columns = col_names
    df.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income'],
      dtype='object')
```

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
[ ] 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       32561 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      32561 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  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
[ ] df.dtypes
```

```
age               int64
workclass        object
fnlwgt            int64
education        object
education_num     int64
marital_status   object
occupation       object
relationship     object
race             object
sex              object
capital_gain      int64
capital_loss      int64
hours_per_week    int64
native_country   object
income           object
dtype: object
```

```
[ ] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	32561.0	38.581647	13.640433	17.0	28.0	37.0	48.0	90.0

```
[ ] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	32561.0	38.581647	13.640433	17.0	28.0	37.0	48.0	90.0
fnlwgt	32561.0	189778.366512	105549.977697	12285.0	117827.0	178356.0	237051.0	1484705.0
education_num	32561.0	10.080679	2.572720	1.0	9.0	10.0	12.0	16.0
capital_gain	32561.0	1077.648844	7385.292085	0.0	0.0	0.0	0.0	99999.0
capital_loss	32561.0	87.303830	402.960219	0.0	0.0	0.0	0.0	4356.0
hours_per_week	32561.0	40.437456	12.347429	1.0	40.0	40.0	45.0	99.0

```
[ ] df.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
age	32561	NaN	NaN	NaN	38.5816	13.6404	17	28	37	48	90
workclass	32561	9	Private	22696	NaN	NaN	NaN	NaN	NaN	NaN	NaN
fnlwgt	32561	NaN	NaN	NaN	189778	105550	12285	117827	178356	237051	1.48470e+06
education	32561	16	HS-grad	10501	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education_num	32561	NaN	NaN	NaN	10.0807	2.57272	1	9	10	12	16
marital_status	32561	7	Married-civ-spouse	14976	NaN	NaN	NaN	NaN	NaN	NaN	NaN
occupation	32561	15	Prof-specialty	4140	NaN	NaN	NaN	NaN	NaN	NaN	NaN
relationship	32561	6	Husband	13193	NaN	NaN	NaN	NaN	NaN	NaN	NaN
race	32561	5	White	27816	NaN	NaN	NaN	NaN	NaN	NaN	NaN
sex	32561	2	Male	21790	NaN	NaN	NaN	NaN	NaN	NaN	NaN
capital_gain	32561	NaN	NaN	NaN	1077.65	7385.29	0	0	0	0	99999
capital_loss	32561	NaN	NaN	NaN	87.3038	402.96	0	0	0	0	4356
hours_per_week	32561	NaN	NaN	NaN	40.4375	12.3474	1	40	40	45	99
native_country	32561	42	United-States	29170	NaN	NaN	NaN	NaN	NaN	NaN	NaN
income	32561	2	<=50K	24720	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
[ ] df.isnull().sum()
```

```
[ ] df.isnull().sum()
```

```
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
```

Исследование котегориальных переменных

```
[ ] categorical = [var for var in df.columns if df[var].dtype=='O']
    df[categorical].head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country	income
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States	<=50K
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K

```
[ ] for var in categorical:
        print(df[var].value_counts()/np.float(len(df)))
```

```
Private             0.697030
Self-emp-not-inc    0.078038
Local-gov           0.064279
?                   0.056386
State-gov           0.039864
Self-emp-inc        0.034274
Federal-gov         0.029483
Without-pay         0.000430
```

```
[ ] for var in categorical:
        print(df[var].value_counts()/np.float(len(df)))
```

```
 Private             0.697030
 Self-emp-not-inc    0.078038
 Local-gov           0.064279
 ?                   0.056386
 State-gov           0.039864
 Self-emp-inc        0.034274
 Federal-gov         0.029483
 Without-pay         0.000430
 Never-worked        0.000215
Name: workclass, dtype: float64
 HS-grad         0.322502
 Some-college    0.223918
 Bachelors       0.164461
 Masters         0.052916
 Assoc-voc       0.042443
 11th            0.036086
 Assoc-acdm      0.032769
 10th            0.028654
 7th-8th         0.019840
 Prof-school     0.017690
 9th             0.015786
 12th            0.013298
 Doctorate       0.012684
 5th-6th         0.010227
 1st-4th         0.005160
 Preschool       0.001566
Name: education, dtype: float64
 Married-civ-spouse       0.459937
 Never-married            0.328092
 Divorced                 0.136452
 Separated                0.031479
 Widowed                  0.030497
 Married-spouse-absent    0.012837
 Married-AF-spouse        0.000706
Name: marital_status, dtype: float64
 Prof-specialty       0.127146
 Craft-repair         0.125887
 Exec-managerial      0.124873
 Adm-clerical         0.115783
 Sales                0.112097
 Other-service        0.101195
 Machine-op-inspct    0.061485
 ?                    0.056601
 Transport-moving     0.049046
 Handlers-cleaners    0.042075
 Farming-fishing      0.030527
 Tech-support         0.028500
```

```
[ ] Name: marital_status, dtype: float64
```

```
 Prof-specialty       0.127146
 Craft-repair         0.125887
 Exec-managerial      0.124873
 Adm-clerical         0.115783
 Sales                0.112097
 Other-service        0.101195
 Machine-op-inspct    0.061485
 ?                    0.056601
 Transport-moving     0.049046
 Handlers-cleaners    0.042075
 Farming-fishing      0.030527
 Tech-support         0.028500
 Protective-serv      0.019932
 Priv-house-serv      0.004576
 Armed-Forces         0.000276
Name: occupation, dtype: float64
 Husband           0.405178
 Not-in-family     0.255060
 Own-child         0.155646
 Unmarried         0.105832
 Wife              0.048156
 Other-relative    0.030128
Name: relationship, dtype: float64
 White                 0.854274
 Black                 0.095943
 Asian-Pac-Islander    0.031909
 Amer-Indian-Eskimo    0.009551
 Other                 0.008323
Name: race, dtype: float64
 Male      0.669205
 Female    0.330795
Name: sex, dtype: float64
 United-States                 0.895857
 Mexico                        0.019748
 ?                             0.017905
 Philippines                   0.006081
 Germany                       0.004207
 Canada                        0.003716
 Puerto-Rico                   0.003501
 El-Salvador                   0.003255
 India                         0.003071
 Cuba                          0.002918
 England                       0.002764
 Jamaica                       0.002488
 South                         0.002457
 China                         0.002303
 Italy                         0.002242
 Dominican-Republic            0.002150
 Vietnam                       0.002058
 Guatemala                     0.001966
```

```
Name: sex, dtype: float64
```

```
United-States                 0.895857
Mexico                        0.019748
?                             0.017905
Philippines                   0.006081
Germany                       0.004207
Canada                        0.003716
Puerto-Rico                   0.003501
El-Salvador                   0.003255
India                         0.003071
Cuba                          0.002918
England                       0.002764
Jamaica                       0.002488
South                         0.002457
China                         0.002303
Italy                         0.002242
Dominican-Republic            0.002150
Vietnam                       0.002058
Guatemala                     0.001966
Japan                         0.001904
Poland                        0.001843
Columbia                      0.001812
Taiwan                        0.001566
Haiti                         0.001351
Iran                          0.001321
Portugal                      0.001136
Nicaragua                     0.001044
Peru                          0.000952
France                        0.000891
Greece                        0.000891
Ecuador                       0.000860
Ireland                       0.000737
Hong                          0.000614
Cambodia                      0.000584
Trinadad&Tobago               0.000584
Laos                          0.000553
Thailand                      0.000553
Yugoslavia                    0.000491
Outlying-US(Guam-USVI-etc)    0.000430
Hungary                       0.000399
Honduras                      0.000399
Scotland                      0.000369
Holand-Netherlands            0.000031
Name: native_country, dtype: float64
 <=50K     0.75919
 >50K      0.24081
Name: income, dtype: float64
```

```
#заменяем фейковые наны на настоящие
df['workclass'].replace(' ?', np.NaN, inplace=True)
df['occupation'].replace(' ?', np.NaN, inplace=True)
```

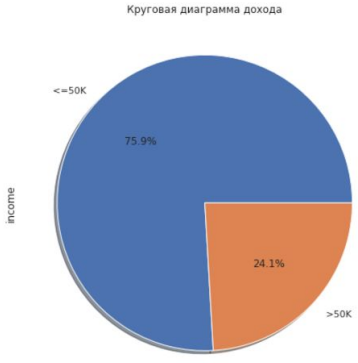
```
>50K      0.24081
Name: income, dtype: float64
```

```
#заменяем фейковые наны на настоящие
df['workclass'].replace(' ?', np.NaN, inplace=True)
df['occupation'].replace(' ?', np.NaN, inplace=True)
df['native_country'].replace(' ?', np.NaN, inplace=True)
```

```
df['income'].unique()
```

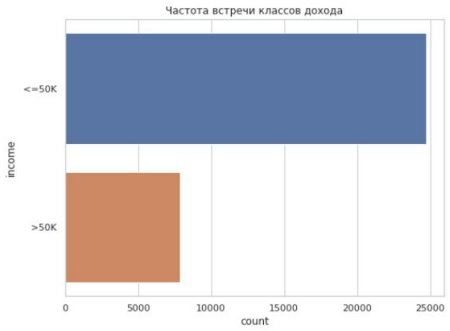
```
array([' <=50K', ' >50K'], dtype=object)
```

```
f,ax=plt.subplots(figsize=(18,8))

ax = df['income'].value_counts().plot.pie(explode=[0,0],autopct='%1.1f%%',ax=ax,shadow=True)
ax.set_title('Круговая диаграмма дохода')

plt.show()
```

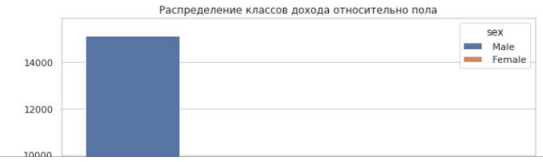


```
[ ]
```

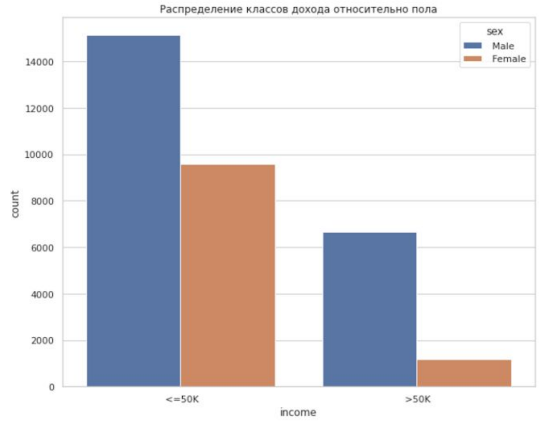
```
[ ] f, ax = plt.subplots(figsize=(8, 6))
    ax = sns.countplot(y="income", data=df)
    ax.set_title("Частота встречи классов дохода")
    plt.show()
```



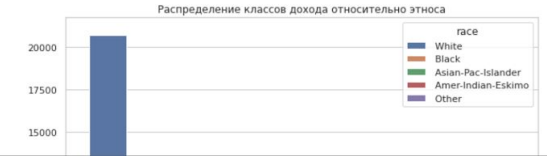

```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.countplot(x="income", hue="sex", data=df)
    ax.set_title("Распределение классов дохода относительно пола")
    plt.show()
```



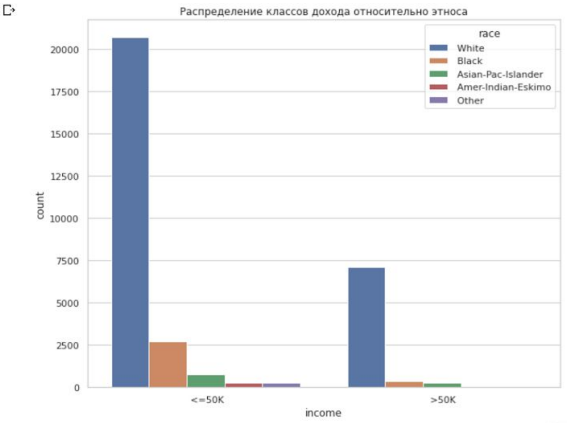

```
ax = sns.countplot(x="income", hue="sex", data=df)
[ ] ax.set_title("Распределение классов дохода относительно пола")
    plt.show()
```




```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.countplot(x="income", hue="race", data=df)
    ax.set_title("Распределение классов дохода относительно этноса")
    plt.show()
```

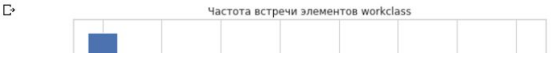



```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.countplot(x="income", hue="race", data=df)
    ax.set_title("Распределение классов дохода относительно этноса")
    plt.show()
```

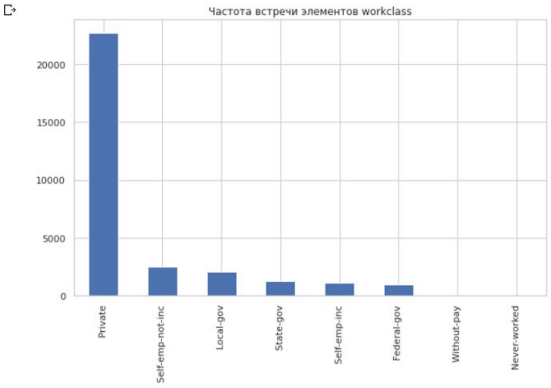


+ Code + Text

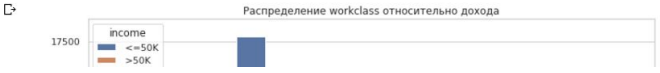
```
[ ] f, ax = plt.subplots(figsize=(10, 6))
    ax = df['workclass'].value_counts().plot(kind="bar")
    ax.set_title("Частота встречи элементов workclass")
    plt.show()
```



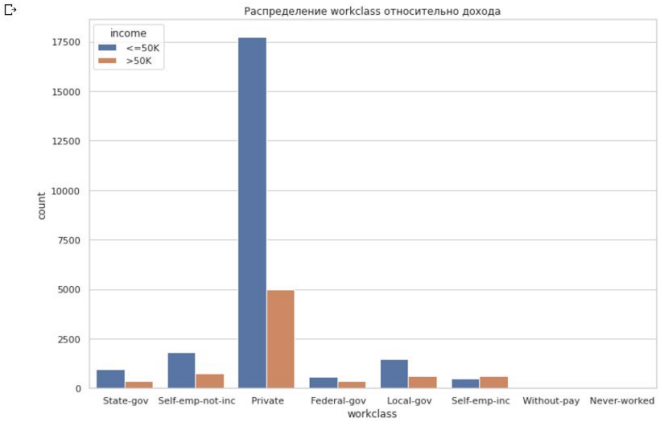

```
[ ] f, ax = plt.subplots(figsize=(10, 6))
    ax = df['workclass'].value_counts().plot(kind="bar")
    ax.set_title("Частота встречи элементов workclass")
    plt.show()
```



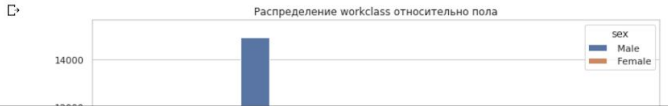

```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="workclass", hue="income", data=df)
    ax.set_title("Распределение workclass относительно дохода")
    plt.show()
```



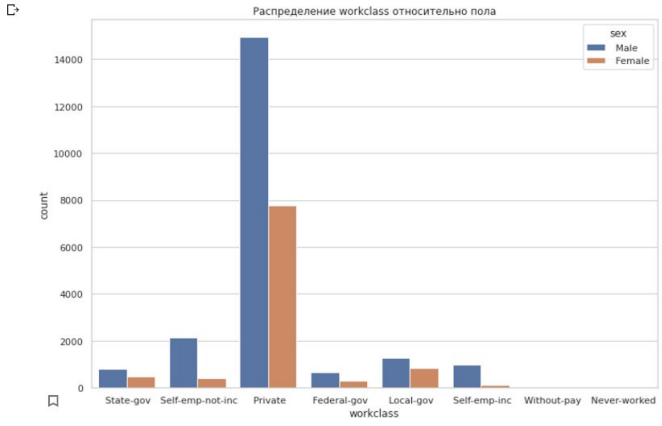
```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="workclass", hue="income", data=df)
    ax.set_title("Распределение workclass относительно дохода")
    plt.show()
```



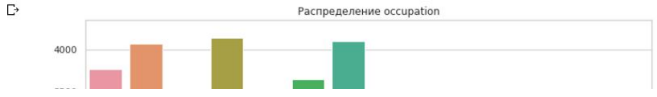

```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="workclass", hue="sex", data=df)
    ax.set_title("Распределение workclass относительно пола")
    plt.show()
```



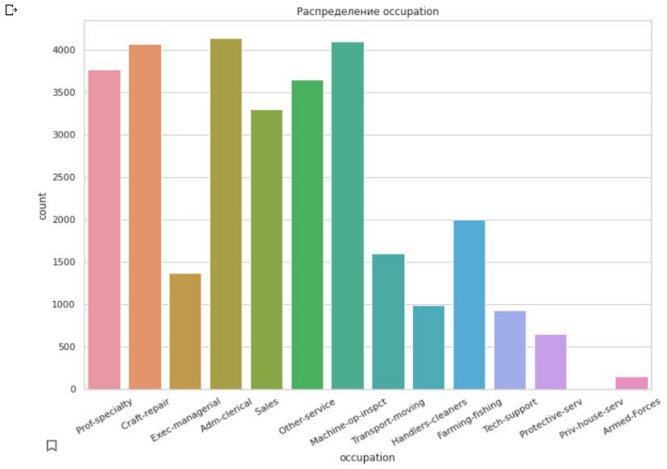

```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="workclass", hue="sex", data=df)
    ax.set_title("Распределение workclass относительно пола")
    plt.show()
```

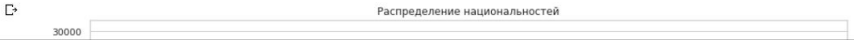

```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="occupation", data=df)
    ax.set_title("Распределение occupation")
    ax.set_xticklabels(df.occupation.value_counts().index, rotation=30)
    plt.show()
```



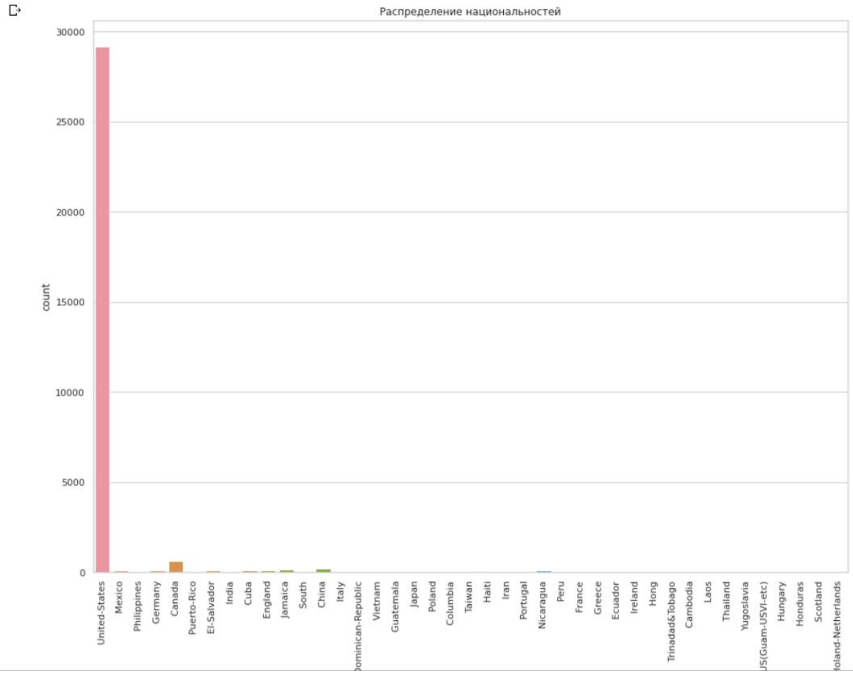
```
[ ] f, ax = plt.subplots(figsize=(12, 8))
    ax = sns.countplot(x="occupation", data=df)
    ax.set_title("Распределение occupation")
    ax.set_xticklabels(df.occupation.value_counts().index, rotation=30)
    plt.show()
```




```
[ ] f, ax = plt.subplots(figsize=(16, 12))
    ax = sns.countplot(x="native_country", data=df)
    ax.set_title("Распределение национальностей")
    ax.set_xticklabels(df.native_country.value_counts().index, rotation=90)
    plt.show()
```




```
    ax = sns.countplot(x="native_country", data=df)
[ ] ax.set_title("Распределение национальностей")
    ax.set_xticklabels(df.native_country.value_counts().index, rotation=90)
    plt.show()
```



Исследование числовых переменных

```
[ ] numerical = [var for var in df.columns if df[var].dtype!='O']
    df[numerical].head()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

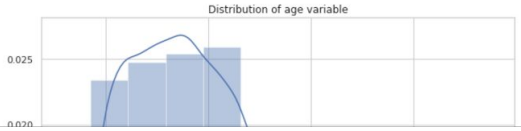
```
[ ] df[numerical].isnull().sum()
```

```
age              0
fnlwgt           0
education_num    0
capital_gain     0
capital_loss     0
hours_per_week   0
dtype: int64
```

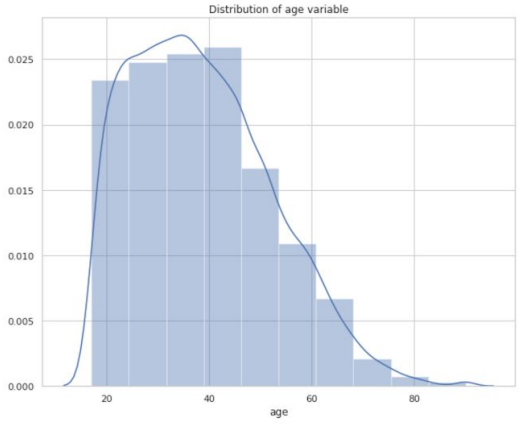
```
[ ] df['age'].nunique()
```

```
73
```

```
[ ] f, ax = plt.subplots(figsize=(10,8))
    x = df['age']
    ax = sns.distplot(x, bins=10)
    ax.set_title("Распределение age")
    plt.show()
```

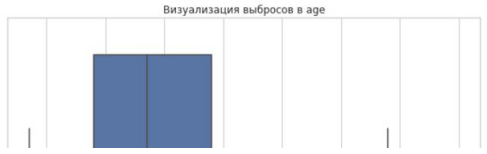



```
[ ] ax.set_title("Распределение age")
    plt.show()
```

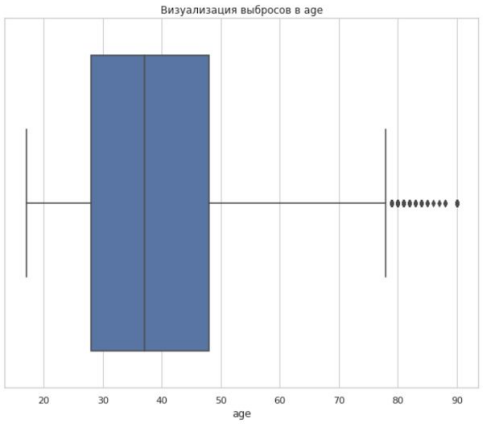


+ Code + Text

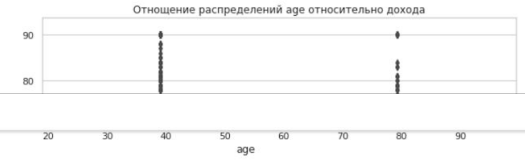
```
[ ] f, ax = plt.subplots(figsize=(10,8))
    x = df['age']
    ax = sns.boxplot(x)
    ax.set_title("Визуализация выбросов в age")
    plt.show()
```



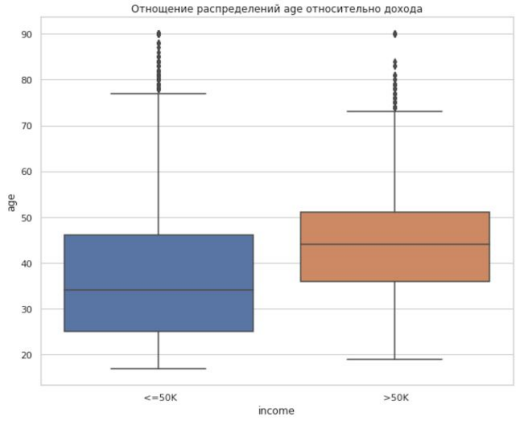
```
[ ] f, ax = plt.subplots(figsize=(10,8))
    x = df['age']
    ax = sns.boxplot(x)
    ax.set_title("Визуализация выбросов в age")
    plt.show()
```



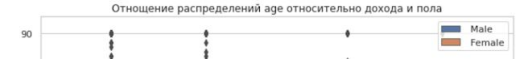

```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.boxplot(x="income", y="age", data=df)
    ax.set_title("Отношение распределений age относительно дохода")
    plt.show()
```



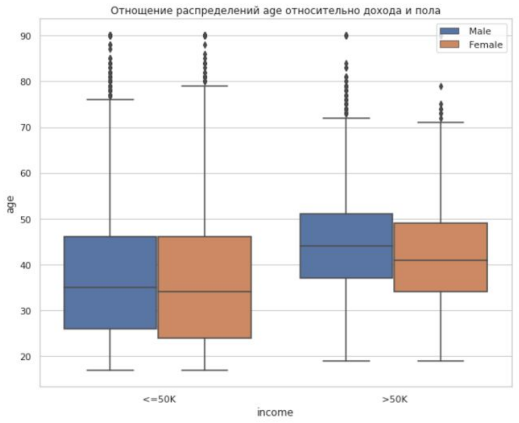

```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.boxplot(x="income", y="age", data=df)
    ax.set_title("Отношение распределений age относительно дохода")
    plt.show()
```




```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.boxplot(x="income", y="age", hue="sex", data=df)
    ax.set_title("Отношение распределений age относительно дохода и пола")
    ax.legend(loc='upper right')
    plt.show()
```

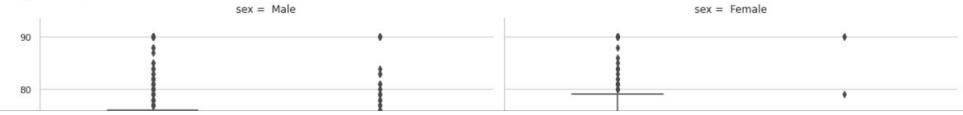


```
[ ] f, ax = plt.subplots(figsize=(10, 8))
    ax = sns.boxplot(x="income", y="age", hue="sex", data=df)
    ax.set_title("Отношение распределений age относительно дохода и пола")
    ax.legend(loc='upper right')
    plt.show()
```



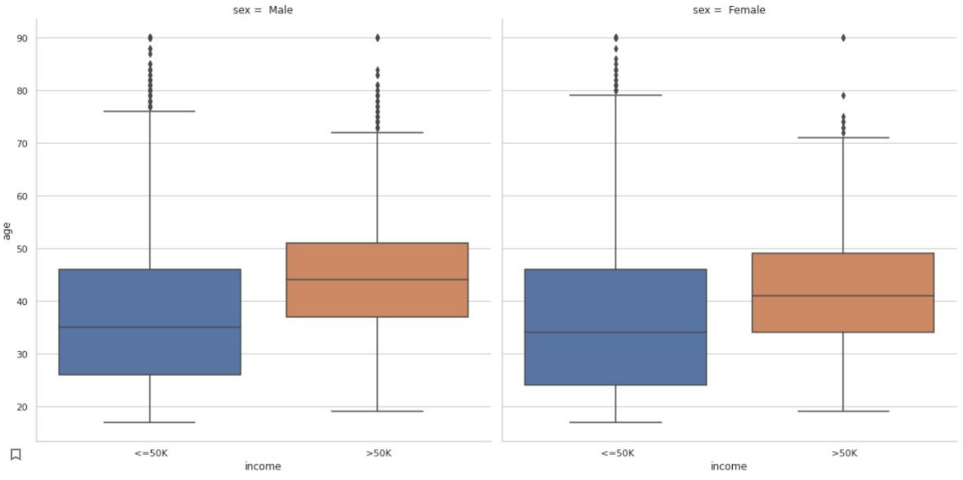

```
[ ] plt.figure(figsize=(8,6))
    ax = sns.catplot(x="income", y="age", col="sex", data=df, kind="box", height=8, aspect=1)
    plt.show()
```

```
<Figure size 576x432 with 0 Axes>
```

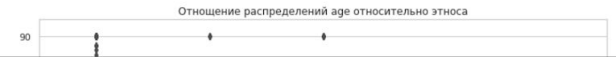



```
[ ] plt.figure(figsize=(8,6))
    ax = sns.catplot(x="income", y="age", col="sex", data=df, kind="box", height=8, aspect=1)
    plt.show()
```

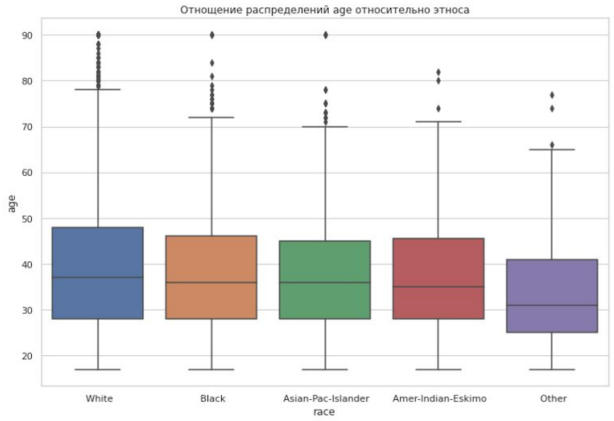
```
<Figure size 576x432 with 0 Axes>
```




```
[ ] plt.figure(figsize=(12,8))
    sns.boxplot(x ='race', y="age", data = df)
    plt.title("Отношение распределений age относительно этноса")
    plt.show()
```



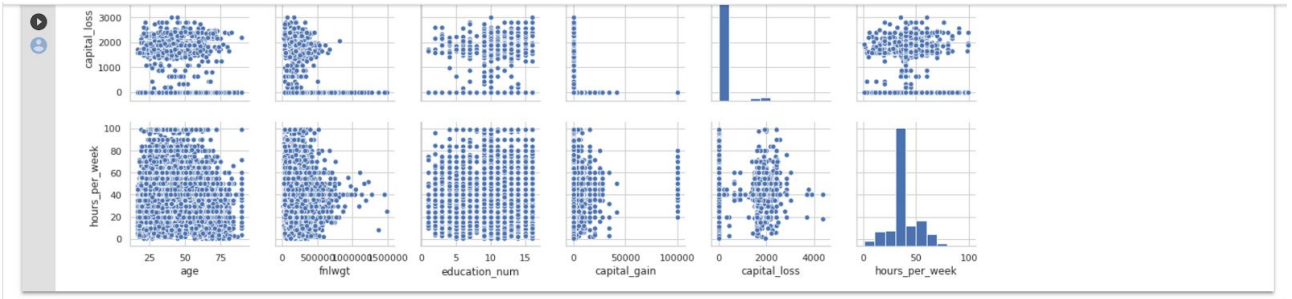
```
sns.boxplot(x ='race', y="age", data = df)
plt.title("Отношение распределений age относительно этноса")
plt.show()
```



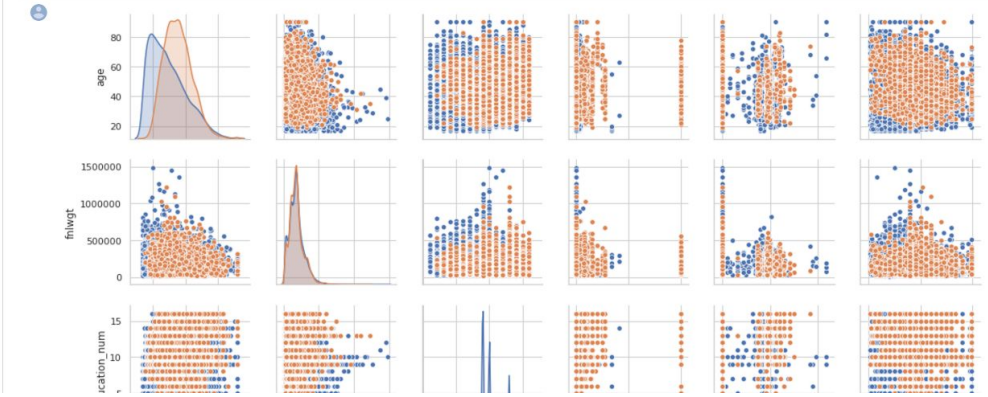

```
#Корреляционная матрица
df.corr().style.format("{:.4}").background_gradient(axis=1)
```

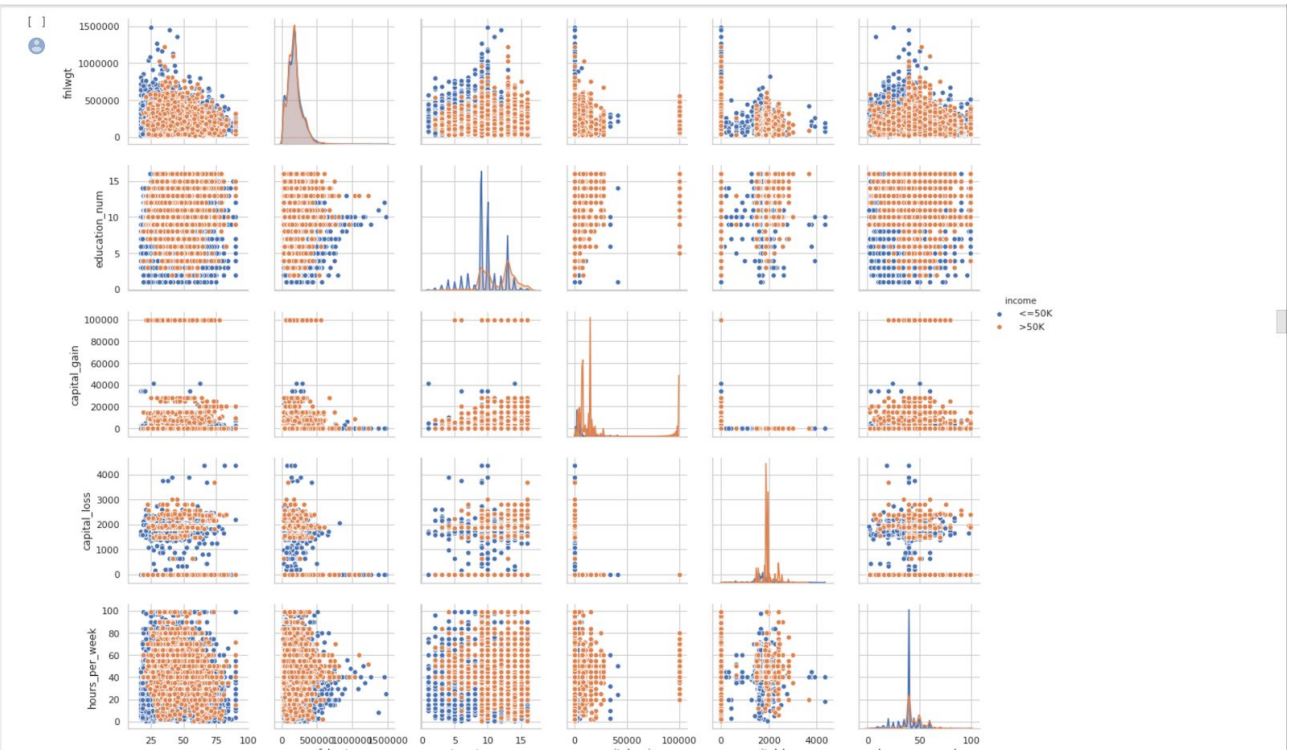
	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
age	1.0	-0.07665	0.03653	0.07767	0.05777	0.06876
fnlwgt	-0.07665	1.0	-0.04319	0.0004319	-0.01025	-0.01877
education_num	0.03653	-0.04319	1.0	0.1226	0.07992	0.1481
capital_gain	0.07767	0.0004319	0.1226	1.0	-0.03162	0.07841
capital_loss	0.05777	-0.01025	0.07992	-0.03162	1.0	0.05426
hours_per_week	0.06876	-0.01877	0.1481	0.07841	0.05426	1.0

```
sns.pairplot(df)
plt.show()
```

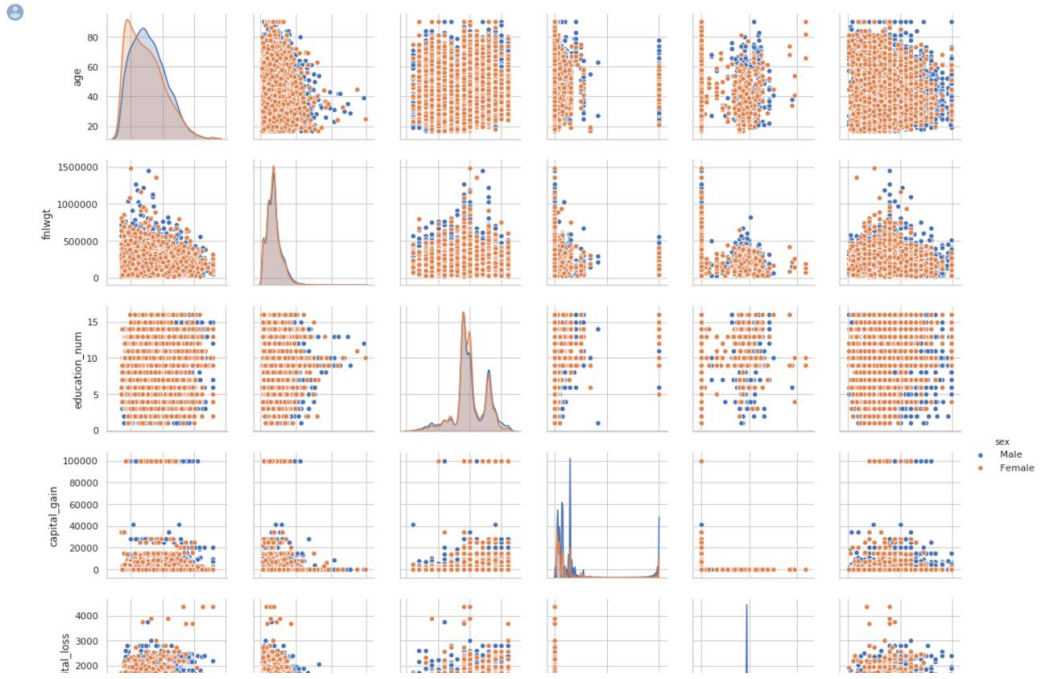


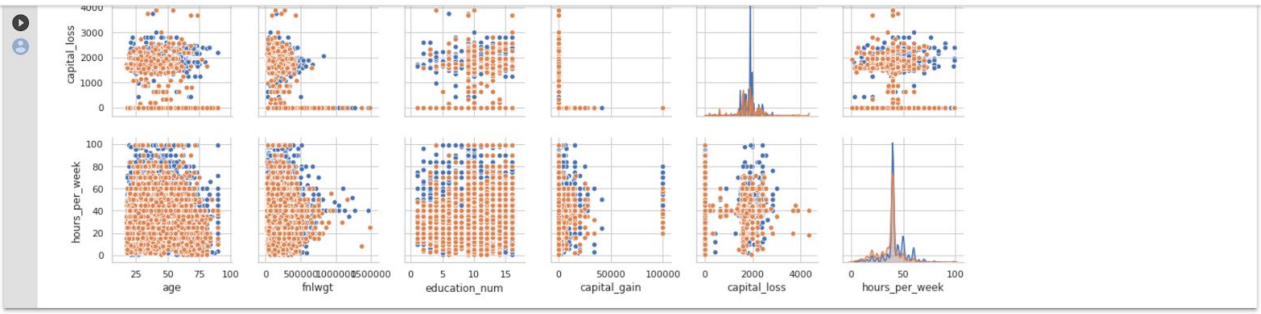

```
[ ] sns.pairplot(df, hue="income")
    plt.show()
```





```
[ ] sns.pairplot(df, hue="sex")
    plt.show()
```





Формирование тестовой и обучеющей выборок

```
[ ] X = df.drop(['income'], axis=1)
    y = df['income']
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
    X_train.shape, X_test.shape
```

```
((22792, 14), (9769, 14))
```

Подготовка данных для моделей

```
[ ] categorical = [col for col in X_train.columns if X_train[col].dtypes == 'O']
    categorical
```

```
['workclass',
 'education',
 'marital_status',
 'occupation',
 'relationship',
 'race',
 'sex',
```

```
[ ] categorical = [col for col in X_train.columns if X_train[col].dtypes == 'O']
    categorical
```

```
['workclass',
 'education',
 'marital_status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native_country']
```

```
[ ] numerical = [col for col in X_train.columns if X_train[col].dtypes != 'O']
    numerical
```

```
['age',
 'fnlwgt',
 'education_num',
 'capital_gain',
 'capital_loss',
 'hours_per_week']
```

```
[ ] X_train[categorical].isnull().mean()
```

```
workclass         0.055985
education         0.000000
marital_status    0.000000
occupation        0.056072
relationship      0.000000
race              0.000000
sex               0.000000
native_country    0.018164
dtype: float64
```

```
[ ] for df2 in [X_train, X_test]:
        df2['workclass'].fillna(X_train['workclass'].mode()[0], inplace=True)
        df2['occupation'].fillna(X_train['occupation'].mode()[0], inplace=True)
        df2['native_country'].fillna(X_train['native_country'].mode()[0], inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py:6245: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._update_inplace(new_data)
```

```
[ ] X_train[categorical].isnull().sum()
```

```
workclass         0
education         0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
native_country    0
dtype: int64
```

```
[ ] X_test[categorical].isnull().sum()
```

```
workclass         0
education         0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
native_country    0
dtype: int64
```

```
[ ] X_train.isnull().sum()
```

```
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
dtype: int64
```

```
[ ] X_test.isnull().sum()
```

```
age               0
workclass         0
```

dtype: int64

```
[ ] X_test.isnull().sum()
```

```
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
dtype: int64
```

Энкодинг признаков

```
[ ] X_train[categorical].head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country
32098	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States
25206	State-gov	HS-grad	Divorced	Adm-clerical	Unmarried	White	Female	United-States
23491	Private	Some-college	Married-civ-spouse	Sales	Husband	White	Male	United-States
12367	Private	HS-grad	Never-married	Craft-repair	Not-in-family	White	Male	Guatemala
7054	Private	7th-8th	Never-married	Craft-repair	Not-in-family	White	Male	Germany

```
[ ] encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'marital_status', 'occupation', 'relationship',
                                     'race', 'sex', 'native_country'])
    X_train = encoder.fit_transform(X_train)
    X_test = encoder.transform(X_test)
```

```
[ ] X_train.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	workclass_8	fnlwgt	education_1	education_2	education_3	education_4
32098	45	1	0	0	0	0	0	0	0	170871	1	0	0	0

```
[ ] X_train.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	workclass_8	fnlwgt	education_1	education_2	education_3	education_4
32098	45	1	0	0	0	0	0	0	0	170871	1	0	0	0
25206	47	0	1	0	0	0	0	0	0	108890	1	0	0	0
23491	48	1	0	0	0	0	0	0	0	187505	0	1	0	0
12367	29	1	0	0	0	0	0	0	0	145592	1	0	0	0
7054	23	1	0	0	0	0	0	0	0	203003	0	0	1	0

5 rows × 105 columns

```
[ ] X_train.shape
```

```
(22792, 105)
```

```
[ ] X_test.shape
```

```
(9769, 105)
```

Масштабирование признаков

```
[ ] cols = X_train.columns
```

```
[ ] scaler = RobustScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
[ ] X_train = pd.DataFrame(X_train, columns=[cols])
```

```
[ ] X_test = pd.DataFrame(X_test, columns=[cols])
```

Обучение моделей и подбор гиперпарамметров

Масштабирование признаков

```
[ ] cols = X_train.columns
```

```
[ ] scaler = RobustScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
[ ] X_train = pd.DataFrame(X_train, columns=[cols])
```

```
[ ] X_test = pd.DataFrame(X_test, columns=[cols])
```

Обучение моделей и подбор гиперпарамметров

```
[ ] pipe.fit(X_train, y_train)
```

```
[ ] lr = LogisticRegressionCV()
    lr.fit(X_train, y_train)
    ans_lr = lr.predict(X_test)
    print(accuracy_score(y_test, ans_lr))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
0.8506500153546934
```

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
ans_dt = dt.predict(X_test)
print(accuracy_score(y_test, ans_dt))
```

```
0.8124680110553792
```

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
ans_rfc = rfc.predict(X_test)
print(accuracy_score(y_test, ans_rfc))
```

```
0.8540280479066434
```

```
cb = CatBoostClassifier(loss_function='Logloss',
                        learning_rate = 0.2,
                        depth = 2,
                        l2_leaf_reg = 3)
cb.fit(X_train, y_train)
ans_cb = cb.predict(X_test)
print(accuracy_score(y_test, ans_cb))
```

```
0:      learn: 0.5556063        total: 10.3ms   remaining: 10.3s
1:      learn: 0.4768272        total: 19.3ms   remaining: 9.63s
2:      learn: 0.4444435        total: 29ms     remaining: 9.63s
3:      learn: 0.4130792        total: 38.3ms   remaining: 9.54s
4:      learn: 0.3960990        total: 47.6ms   remaining: 9.48s
5:      learn: 0.3856152        total: 57.9ms   remaining: 9.6s
6:      learn: 0.3722368        total: 67ms     remaining: 9.51s
7:      learn: 0.3662939        total: 77ms     remaining: 9.55s
8:      learn: 0.3593511        total: 86.2ms   remaining: 9.5s
9:      learn: 0.3545251        total: 94.7ms   remaining: 9.37s
10:     learn: 0.3484984        total: 103ms    remaining: 9.28s
11:     learn: 0.3457898        total: 114ms    remaining: 9.35s
12:     learn: 0.3427428        total: 123ms    remaining: 9.3s
13:     learn: 0.3393218        total: 132ms    remaining: 9.27s
14:     learn: 0.3361214        total: 141ms    remaining: 9.24s
15:     learn: 0.3344353        total: 149ms    remaining: 9.16s
16:     learn: 0.3316364        total: 158ms    remaining: 9.13s
17:     learn: 0.3294597        total: 167ms    remaining: 9.09s
18:     learn: 0.3273995        total: 175ms    remaining: 9.05s
19:     learn: 0.3256939        total: 186ms    remaining: 9.1s
```

```
981:    learn: 0.2510254        total: 10s      remaining: 184ms
982:    learn: 0.2510155        total: 10s      remaining: 173ms
983:    learn: 0.2509996        total: 10s      remaining: 163ms
984:    learn: 0.2509948        total: 10s      remaining: 153ms
985:    learn: 0.2509831        total: 10.1s    remaining: 143ms
986:    learn: 0.2509807        total: 10.1s    remaining: 133ms
987:    learn: 0.2509659        total: 10.1s    remaining: 122ms
988:    learn: 0.2509564        total: 10.1s    remaining: 112ms
989:    learn: 0.2509400        total: 10.1s    remaining: 102ms
990:    learn: 0.2509324        total: 10.1s    remaining: 91.8ms
991:    learn: 0.2509204        total: 10.1s    remaining: 81.6ms
992:    learn: 0.2509079        total: 10.1s    remaining: 71.4ms
993:    learn: 0.2508897        total: 10.1s    remaining: 61.2ms
994:    learn: 0.2508785        total: 10.1s    remaining: 51ms
995:    learn: 0.2508666        total: 10.2s    remaining: 40.8ms
996:    learn: 0.2508432        total: 10.2s    remaining: 30.6ms
997:    learn: 0.2508046        total: 10.2s    remaining: 20.4ms
998:    learn: 0.2507860        total: 10.2s    remaining: 10.2ms
999:    learn: 0.2507752        total: 10.2s    remaining: 0us
0.8734773262360528
```

```
svc = SVC()
svc.fit(X_train, y_train)
ans_svc = svc.predict(X_test)
print(accuracy_score(y_test, ans_svc))
```

```
0.8029481011362473
```

Вывод

В данном проекте лучшей моделью оказалась catboost classifier с подобраными гипер параметрами, считаю что для "голой" модели это хороший результат.

Заключение

В данном курсовом проекте была решена типовая задача машинного обучения. Был выбран набор данных для построения моделей машинного обучения, проведен разведочный анализ данных и построены графики, необходимые для понимания структуры данных. Были выбраны признаки, подходящие для построения моделей, масштабированы данные и проведен корреляционный анализ данных. Это позволило сформировать промежуточные выводы о возможности построения моделей машинного обучения.

На следующем этапе были выбраны метрики для последующей оценки качества моделей и наиболее подходящие модели для решения задачи классификации. Затем были сформированы обучающая и тестовая выборки на основе исходного набора данных и построено базовое решение для выбранных моделей без подбора гиперпараметров.

Следующим шагом был подбор гиперпараметров для выбранных моделей, после чего мы смогли сравнить качество полученных моделей с качеством baseline-моделей. Большинство моделей, для которых были подобраны оптимальные значения гиперпараметров, показали лучший результат.

В заключение, были сформированы выводы о качестве построенных моделей на основе выбранных метрик. Для наглядности результаты сравнения качества были отображены в виде графиков, а также сделаны выводы в форме текстового описания. Четыре метрики показали, что для выбранного набора данных лучшей моделью оказалась «машина опорных векторов».

Список использованных источников

1. Ю.Е. Гапанюк, Лекции по курсу «Технологии машинного обучения» 2019-2020 учебный год.
2. scikit-learn Machine Learning in Python: [сайт]. URL: https://scikit-learn.org/stable/
3. Income Classification [Электронный ресурс]. URL: https://www.kaggle.com/lodetomasi1995/income-classification (дата обращения: 24.05.2020)