Анализ удовлетворённости сотрудников и предсказание оттока персонала

Введение

В современном мире, где данные играют ключевую роль в принятии решений, HR-аналитика становится неотъемлемой частью успешного управления персоналом. Компания "Работа с заботой" стремится использовать мощь данных для оптимизации HR-процессов, чтобы минимизировать финансовые потери и снизить отток ценных сотрудников. В этом проекте мы сосредоточимся на двух основных задачах: предсказании уровня удовлетворённости сотрудников и вероятности их увольнения.

Удовлетворённость сотрудников — это не просто показатель благополучия рабочего коллектива, но и важный фактор, влияющий на общую производительность и стабильность компании. Понимание того, что делает сотрудников довольными или недовольными, позволяет предпринимать целенаправленные действия для улучшения рабочей среды и политики компании.

Предсказание оттока — ключевая задача для HR-аналитиков, поскольку внезапные увольнения могут нанести ущерб бизнес-процессам и повлечь за собой значительные финансовые потери. Разработка модели, способной предсказать уход сотрудника, дает компании возможность предотвратить такие ситуации, предпринимая профилактические меры.

В рамках данного проекта мы будем использовать данные, предоставленные компанией, включая результаты опросов удовлетворённости, для создания предиктивных моделей машинного обучения. Эти модели помогут нам не только понять текущее состояние дел в компании, но и предвидеть будущие тенденции, связанные с удовлетворённостью и оттоком персонала.

Загрузка данных

Библиотеки

```
!pip install scikit-learn==1.4.0 -q
!pip install matplotlib==3.8.4 -q
!pip install numpy==1.21.1 -q
!pip install seaborn==0.13.2 -q

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import train_test_split
```

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder,
StandardScaler, MinMaxScaler, TargetEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, make_scorer,
roc_auc_score
import seaborn as sns
from sklearn.svm import SVC
```

Загрузка исходных данных

```
train_job = pd.read_csv('/datasets/train_job_satisfaction_rate.csv')
#Тренировочная выборка
test_features = pd.read_csv('/datasets/test_features.csv') #Входные
признаки тестовой выборки
test_job =
pd.read_csv('/datasets/test_target_job_satisfaction_rate.csv')
#Целевой признак тестовой выборки
```

Первичное ознакомление с данными

Для удобства создадим словарь датафреймов.

```
dfs = {'train_job':train_job, 'test_features':test_features,
      'test_job':test_job}

for df in dfs:
    print(df)
    display(dfs[df].head(),dfs[df].info())
```

```
train_job
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     4000 non-null   int64
 1   dept                   3994 non-null   object
 2   level                  3996 non-null   object
 3   workload               4000 non-null   object
 4   employment_years       4000 non-null   int64
 5   last_year_promo        4000 non-null   object
 6   last_year_violations   4000 non-null   object
 7   supervisor_evaluation  4000 non-null   int64
 8   salary                 4000 non-null   int64
```

```
 9   job_satisfaction_rate  4000 non-null   float64
dtypes: float64(1), int64(4), object(5)
memory usage: 312.6+ KB

       id        dept   level workload  employment_years
last_year_promo  \
0  155278       sales  junior   medium                 2
no
1  653870          hr  junior     high                 2
no
2  184592       sales  junior      low                 1
no
3  171431  technology  junior      low                 4
no
4  693419          hr  junior   medium                 1
no

  last_year_violations  supervisor_evaluation  salary
job_satisfaction_rate
0                   no                     1   24000
0.58
1                   no                     5   38400
0.76
2                   no                     2   12000
0.11
3                   no                     2   18000
0.37
4                   no                     3   22800
0.20

None

test_features
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    2000 non-null   int64
 1   dept                  1998 non-null   object
 2   level                 1999 non-null   object
 3   workload              2000 non-null   object
 4   employment_years      2000 non-null   int64
 5   last_year_promo       2000 non-null   object
 6   last_year_violations  2000 non-null   object
 7   supervisor_evaluation  2000 non-null   int64
 8   salary                2000 non-null   int64
dtypes: int64(4), object(5)
memory usage: 140.8+ KB
```

```
       id       dept   level workload  employment_years
last_year_promo  \
0  485046  marketing  junior   medium                 2
no
1  686555         hr  junior   medium                 1
no
2  467458      sales  middle      low                 5
no
3  418655      sales  middle      low                 6
no
4  789145         hr  middle   medium                 5
no

  last_year_violations  supervisor_evaluation  salary
0                   no                     5   28800
1                   no                     4   30000
2                   no                     4   19200
3                   no                     4   19200
4                   no                     5   40800

None

test_job
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     2000 non-null   int64
 1   job_satisfaction_rate  2000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 31.4 KB

       id  job_satisfaction_rate
0  130604                   0.74
1  825977                   0.75
2  418490                   0.60
3  555320                   0.72
4  826430                   0.08

None
```

В нашем распоряжении 4000 тренировочных и 2000 тестовых строк, в некоторых из них имеются пропуски.

Предобработка данных

Пропуски заполним позже в пайплайне

```
for df in dfs:
    print(df,dfs[df].duplicated().sum())
```

```
train_job 0
test_features 0
test_job 0
```

Явных дубликатов нет

```
for df in dfs:
    for column in dfs[df]:
        if dfs[df][column].dtype==object:
            display(dfs[df][column].unique())
```

```
array(['sales', 'hr', 'technology', 'purchasing', 'marketing', nan],
      dtype=object)
```

```
array(['junior', 'middle', 'sinior', nan], dtype=object)
```

```
array(['medium', 'high', 'low'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['marketing', 'hr', 'sales', 'purchasing', 'technology', nan, '
'],
      dtype=object)
```

```
array(['junior', 'middle', 'sinior', nan], dtype=object)
```

```
array(['medium', 'low', 'high', ' '], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
for df in dfs:
    dfs[df].replace(to_replace= {' ':np.nan},inplace = True)
```

Пустое значение было заменено np.nan, но это можно было сделать и чуть позже пайплайне

Исследовательский анализ данных

Исследовательский анализ данных начинается с описательной статистики, которая включает в себя расчет основных статистических показателей, таких как среднее значение, медиана, мода, минимум, максимум и стандартное отклонение для количественных переменных. Для категориальных переменных мы рассмотрим частоту встречаемости различных категорий. Это дает нам первое представление о распределении данных и возможных аномалиях, таких как выбросы или пропущенные значения.

Затем мы переходим к визуализации данных, используя графики и диаграммы, такие как гистограммы, ящики с усами (box plots), точечные диаграммы (scatter plots) и тепловые карты корреляций, чтобы увидеть взаимосвязи между переменными и выявить закономерности или аномалии.

После этого мы проведем анализ пропущенных значений и решим, как лучше их обработать — удалить, заменить на среднее/медиану или использовать дургие методы.

Также важно исследовать категориальные переменные и принять решение о том, как их кодировать для использования в моделях машинного обучения. Мы можем использовать подходы, такие как One-Hot Encoding или Label Encoding, в зависимости от типа и количества категорий.

Наконец, мы оценим необходимость масштабирования признаков, особенно если мы планируем использовать алгоритмы, чувствительные к масштабу, такие как SVM или KNN.

Статистический анализ признаков

```
for df in dfs:
    display(dfs[df].drop('id',axis=1).describe())
```

```
       employment_years  supervisor_evaluation        salary  \
count       4000.000000            4000.000000   4000.000000
mean           3.718500               3.476500  33926.700000
std            2.542513               1.008812  14900.703838
min            1.000000               1.000000  12000.000000
25%            2.000000               3.000000  22800.000000
50%            3.000000               4.000000  30000.000000
75%            6.000000               4.000000  43200.000000
max           10.000000               5.000000  98400.000000

       job_satisfaction_rate
count            4000.000000
mean                0.533995
std                 0.225327
min                 0.030000
25%                 0.360000
50%                 0.560000
75%                 0.710000
max                 1.000000

       employment_years  supervisor_evaluation        salary
count       2000.000000            2000.000000   2000.000000
mean           3.666500               3.526500  34066.800000
std            2.537222               0.996892  15398.436729
min            1.000000               1.000000  12000.000000
25%            1.000000               3.000000  22800.000000
50%            3.000000               4.000000  30000.000000
75%            6.000000               4.000000  43200.000000
max           10.000000               5.000000  96000.000000
```

```
       job_satisfaction_rate
count             2000.00000
mean                 0.54878
std                  0.22011
min                  0.03000
25%                  0.38000
50%                  0.58000
75%                  0.72000
max                  1.00000
```

```
for df in dfs:
    display(dfs[df].drop('id',axis=1).describe())
```

```
       employment_years  supervisor_evaluation        salary  \
count       4000.000000            4000.000000   4000.000000
mean           3.718500               3.476500  33926.700000
std            2.542513               1.008812  14900.703838
min            1.000000               1.000000  12000.000000
25%            2.000000               3.000000  22800.000000
50%            3.000000               4.000000  30000.000000
75%            6.000000               4.000000  43200.000000
max           10.000000               5.000000  98400.000000
```

```
       job_satisfaction_rate
count            4000.000000
mean                0.533995
std                 0.225327
min                 0.030000
25%                 0.360000
50%                 0.560000
75%                 0.710000
max                 1.000000
```

```
       employment_years  supervisor_evaluation        salary
count       2000.000000            2000.000000   2000.000000
mean           3.666500               3.526500  34066.800000
std            2.537222               0.996892  15398.436729
min            1.000000               1.000000  12000.000000
25%            1.000000               3.000000  22800.000000
50%            3.000000               4.000000  30000.000000
75%            6.000000               4.000000  43200.000000
max           10.000000               5.000000  96000.000000
```

```
       job_satisfaction_rate
count             2000.00000
mean                 0.54878
std                  0.22011
min                  0.03000
25%                  0.38000
50%                  0.58000
```

```
75%                   0.72000
max                   1.00000
```

Выбросов нет, значения в тестовой выборке и в тренировочной распределены похоже

```
for df in dfs:
    for column in dfs[df]:
        if dfs[df][column].dtype==object:
            display(column, dfs[df][column].value_counts())
```

```
'dept'
```

```
sales         1512
technology     866
purchasing     610
marketing      550
hr             456
Name: dept, dtype: int64
```

```
'level'
```

```
junior    1894
middle    1744
sinior     358
Name: level, dtype: int64
```

```
'workload'
```

```
medium    2066
low       1200
high       734
Name: workload, dtype: int64
```

```
'last_year_promo'
```

```
no     3880
yes     120
Name: last_year_promo, dtype: int64
```

```
'last_year_violations'
```

```
no     3441
yes     559
Name: last_year_violations, dtype: int64
```

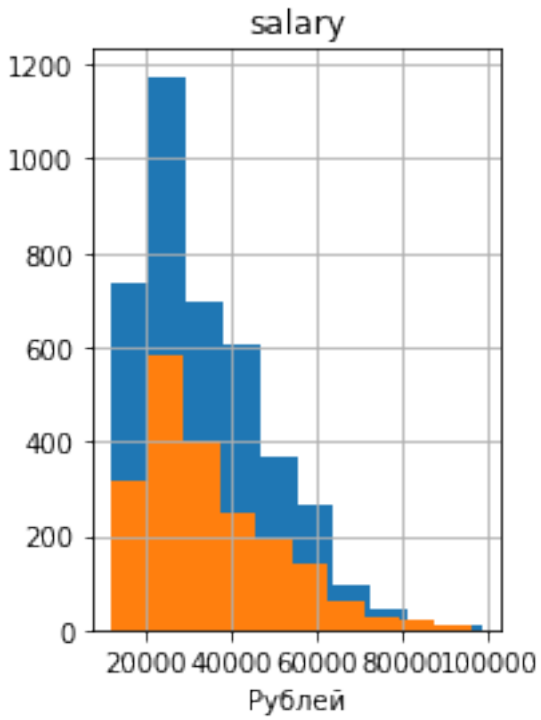
```
'dept'
```

```
sales         763
technology    455
marketing     279
purchasing    273
```

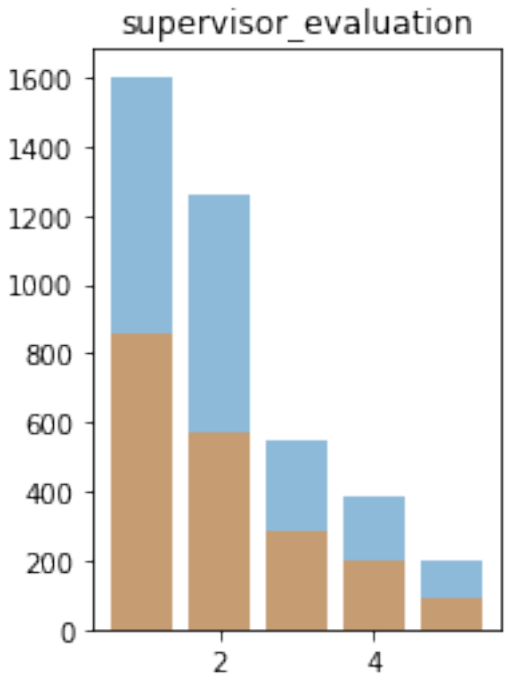
```
hr            227
Name: dept, dtype: int64

'level'

junior    974
middle    854
sinior    171
Name: level, dtype: int64

'workload'

medium    1043
low        593
high       363
Name: workload, dtype: int64

'last_year_promo'

no     1937
yes      63
Name: last_year_promo, dtype: int64

'last_year_violations'

no     1738
yes     262
Name: last_year_violations, dtype: int64
```

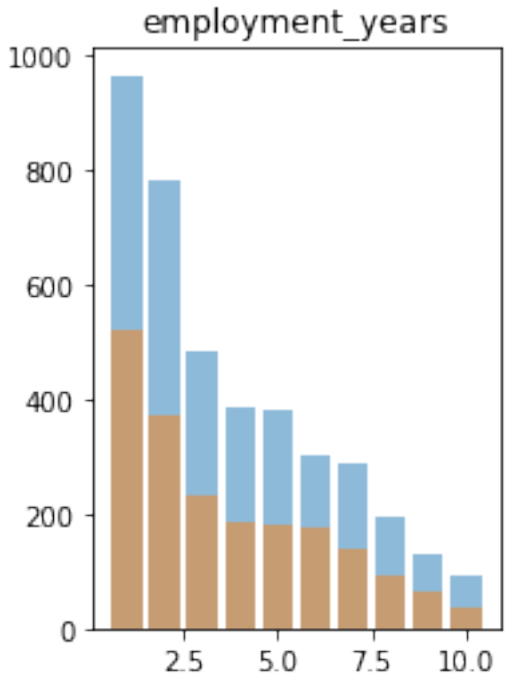
Остается посмотреть на гистограммы и круговые диаграммы.

```
plt.subplot(1, 2, 1)
dfs['train_job']['salary'].hist()
plt.title('salary')
plt.xlabel('Рублей')
plt.subplot(1, 2, 1)
dfs['test_features']['salary'].hist()
plt.title('salary')
plt.xlabel('Рублей')
plt.show();
```




```
for column in ['supervisor_evaluation','employment_years']:
    plt.subplot(1, 2, 1)
    plt.bar(dfs['train_job']
[column].sort_values().unique(),dfs['train_job']
[column].value_counts(),alpha=0.5)
    plt.title(column)
    plt.subplot(1, 2, 1)
    plt.bar(dfs['test_features']
[column].sort_values().unique(),dfs['test_features']
[column].value_counts(),alpha=0.5)
    plt.title(column)
    plt.show();
```

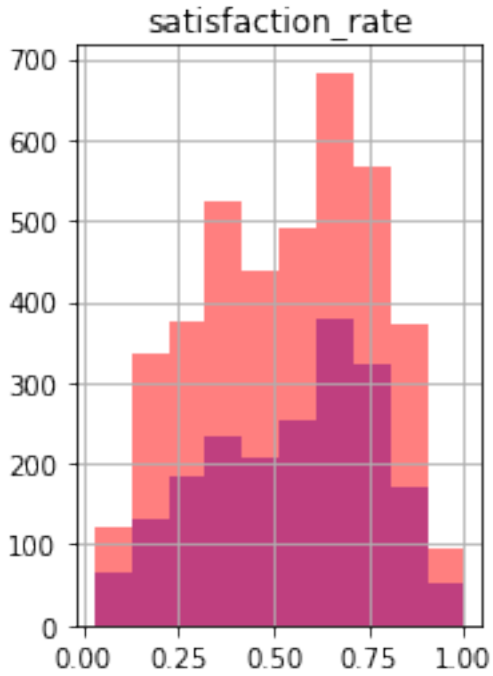




Гистограммы количественных признаков имеют не нормальное распределение, посмотрим на диаграмму целевого признака.

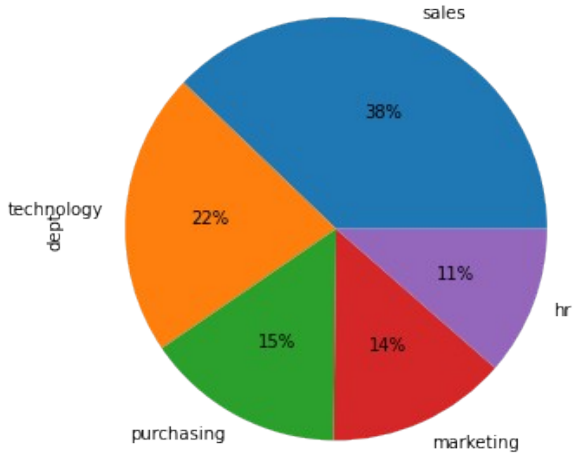
```
plt.subplot(1, 2, 2)
dfs['test_job']['job_satisfaction_rate'].hist(alpha=0.5, color='blue')
plt.title('satisfaction_rate')
plt.subplot(1, 2, 2)
```

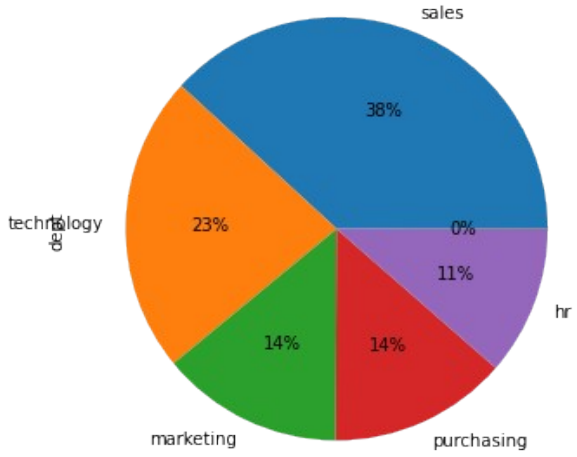
```
dfs['train_job']['job_satisfaction_rate'].hist(alpha=0.5, color='red')
plt.title('satisfaction_rate')
plt.show();
```

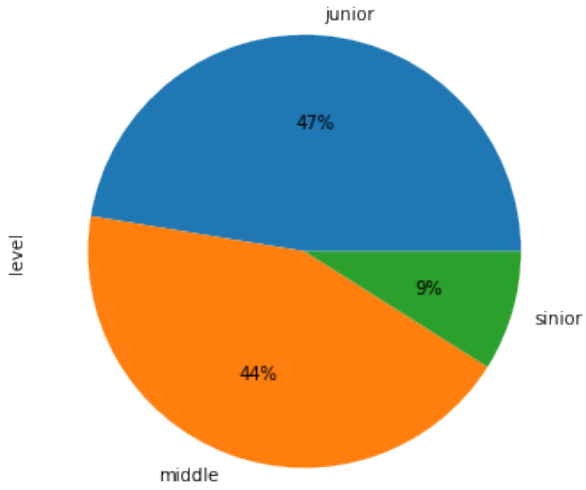


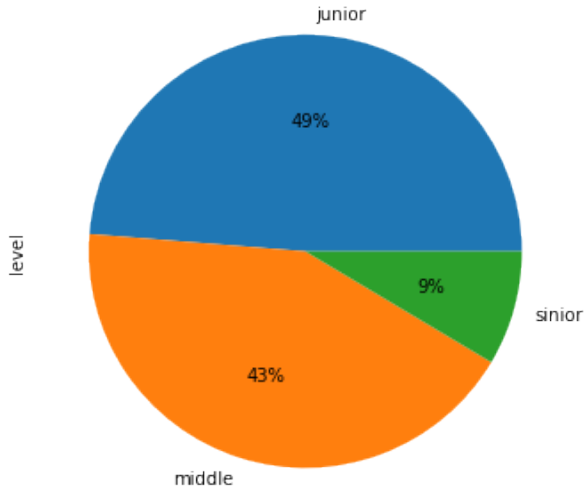
Распределение скошено вправо, большая часть работников удовлетворено совей работой.

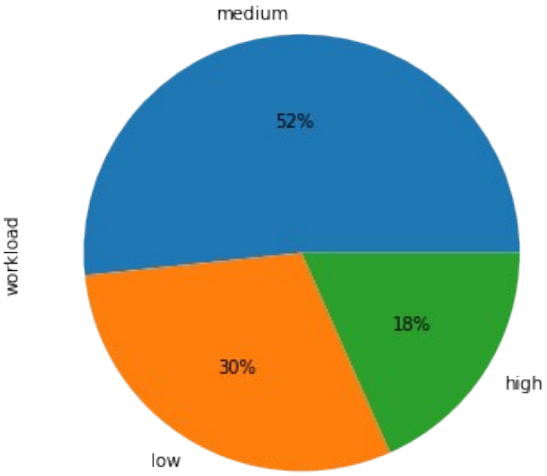
```
for column in dfs['test_features']:
    if dfs['test_features'][column].dtype==object:
        plt.subplot(1, 2, 1)
        dfs['train_job']
[column].value_counts().plot.pie(figsize=(12,12),autopct='%1.f%%')
        plt.subplot(1, 2, 2)
        dfs['test_features']
[column].value_counts().plot.pie(figsize=(12,12),autopct='%1.f%%')
        plt.show();
```

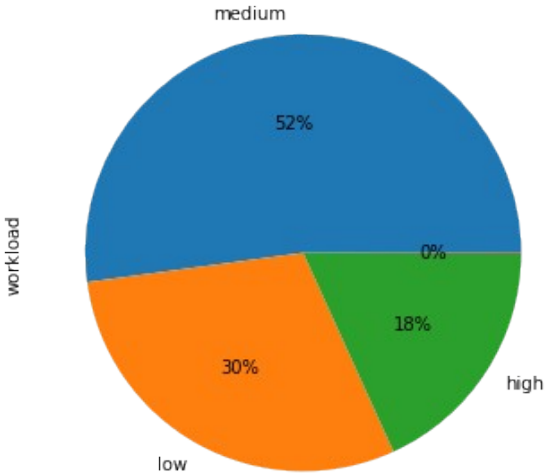

sales
38%
technology
22%
dept
11%
hr
15%
14%
purchasing
marketing

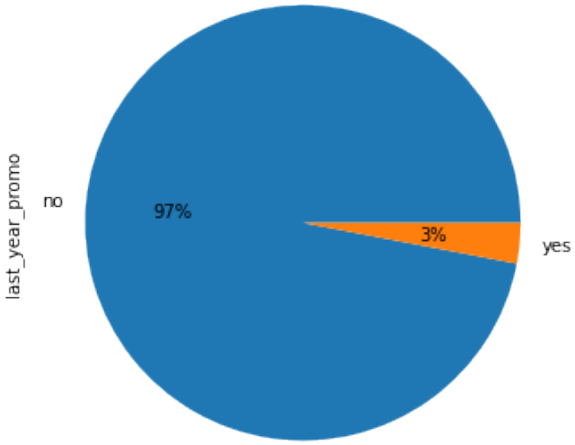

sales
38%
technology
23%
dept
0%
11%
hr
14%
14%
marketing
purchasing

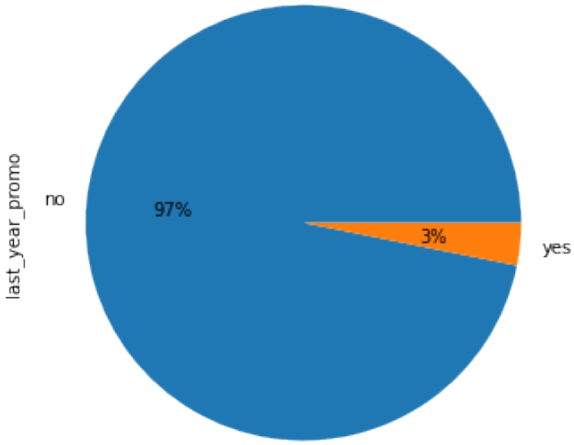

junior
47%
level
9%
sinior
44%
middle

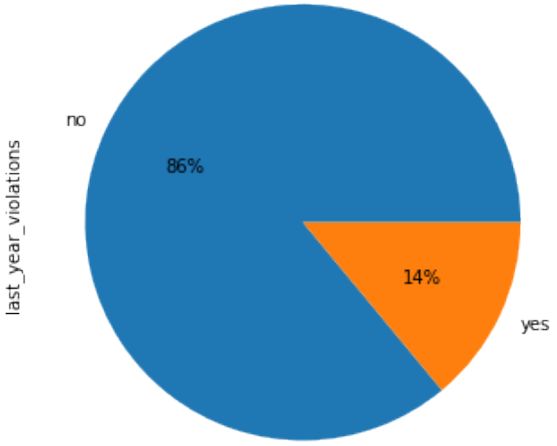

junior
49%
level
9%
sinior
43%
middle

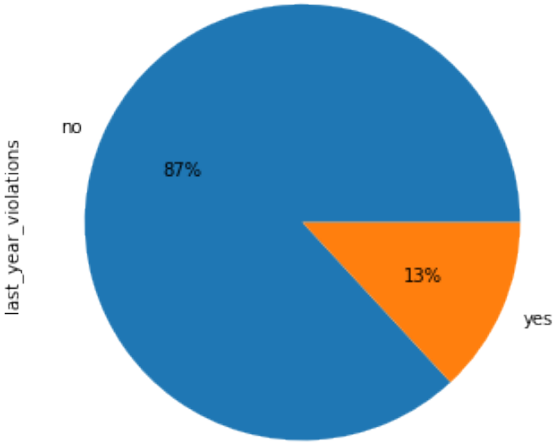

medium
52%
workload
18%
30%
high
low


medium
52%
workload
0%
18%
30%
high
low


last_year_promo
no
97%
3%
yes


last_year_promo
no
97%
3%
yes





Исходя из предоставленных данных, можно сделать несколько ключевых наблюдений:

Годы работы в компании: Большая часть сотрудников работает в компании уже 3 года, Стандартное отклонение равно 2.54, что указывает на разнообразие в опыте работы сотрудников, при этом минимальный стаж работы составляет 1 год, а максимальный — 10 лет.

Оценка руководителем: Средняя оценка руководителем составляет 3.48, стандартное отклонение равно 1.01, что может свидетельствовать о различиях в оценках между руководителями или отделами.

Зарплата: Средняя зарплата сотрудников составляет 30 000 рублей, стандартное отклонение в размере 14,900.70 рублей указывает на значительные различия в оплате труда.

Распределение по отделам: Большинство сотрудников работают в отделах продаж и технологий. Это может указывать на то, что эти отделы являются основными для бизнеса компании. Возможно, стоит уделить особое внимание удовлетворенности сотрудников именно в этих отделах, так как они могут иметь наибольшее влияние на общую производительность.

Уровень должности: Большинство сотрудников находятся на начальном и среднем уровне. Сотрудников старшего уровня значительно меньше, что может говорить о пирамидальной структуре управления или о возможных проблемах с карьерным ростом в компании.

Рабочая нагрузка: Сотрудники с умеренной рабочей нагрузкой составляют большинство. Важно убедиться, что распределение рабочей нагрузки соответствует возможностям и предпочтениям сотрудников, чтобы избежать перегрузок и снижения удовлетворенности.

Продвижение по службе в прошлом году: Очень мало сотрудников получили повышение в прошлом году. Это может быть потенциальным фактором недовольства и может способствовать увеличению оттока персонала.

Нарушения в прошлом году: Большинство сотрудников не имели нарушений в прошлом году, что может указывать на хорошую дисциплину или эффективную политику управления персоналом.

Уровень удовлетворенности работой: Средний уровень удовлетворенности работой составляет 0.534. Стандартное отклонение равно 0.225, что показывает разнообразие уровней удовлетворенности среди сотрудников.

Корреляционный анализ

Не все значения распределены нормально воспользуемся корреляцией спирмена

```
dfs['train_job'].drop('id',axis=1).corr(method =
'spearman').style.background_gradient(cmap='coolwarm')
```

```
<pandas.io.formats.style.Styler at 0x7f6a5fe081c0>
```

- Коэффициент корреляции 0.472688 свидетельствует о средней положительной связи между **стажем работы и зарплатой**, это означает, что с увеличением стажа зарплата сотрудников также растет.
- Коэффициент корреляции 0.218589 указывает на слабую положительную связь между **стажем работы и удовлетворенностью работой**.
- Довольно высокий коэффициент корреляции 0.746608, указывает на сильную положительную связь между **оценкой руководителем и удовлетворенностью работой**. Это может указывать на то, что качество взаимоотношений между начальником и подчиненным оказывает значительное влияние на уровень удовлетворенности работой. Это предполагает, что положительное восприятие руководства и поддержка со стороны начальства могут способствовать более высокому уровню удовлетворенности сотрудников.

  Такие результаты подчеркивают важность эффективного управления и хороших отношений в рабочей среде. Они также могут служить основанием для разработки программ по улучшению коммуникации и взаимодействия между руководителями и их командами, что в конечном итоге может привести к улучшению общей производительности и удовлетворенности на работе.
- Коэффициент корреляции 0.126707 является довольно низким, что говорит о слабой положительной связи между **зарплатой и удовлетворенностью работой**. Возможно стоит сосредоточится на улучшении условий труда а также рассмотреть иные способы мотивации сотрудников.

Важно отметить, что корреляция не обязательно указывает на причинно-следственную связь, а только на степень линейной связи между переменными. Для более глубокого понимания взаимосвязей между этими переменными может потребоваться дополнительный анализ.

Подготовка данных и создание моделей

Обучим 2 модели с различным гиперпараметрами, для этого используем один общий пайплайн для всех моделей и инструмент подбора гиперпараметров, который вернёт лучшую модель, а также выполним замену пропусков на наиболее частое значение.

```
train_job = dfs['train_job'].drop('id',axis=1)

def smape(y_true, y_pred):
    return 100 / len(y_true) * np.sum(2 * np.abs(y_true - y_pred) /
((np.abs(y_true) + np.abs(y_pred))))

smape_score = make_scorer(smape, greater_is_better=False)

RANDOM_STATE = 42
TEST_SIZE = 0.25

#Создаем списки признаков
ohe_col = ['dept', 'last_year_promo','last_year_violations']
num_col =
train_job.select_dtypes(exclude=['object']).columns.drop('job_satisfac
tion_rate')
ord_col = ['level','workload']

ohe_pipe = Pipeline(
    [('simpleImputer_ohe', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ohe', OneHotEncoder(drop='first', handle_unknown='ignore',
sparse_output=False))
    ]
    )

# создаём пайплайн для подготовки признаков из списка ord_columns:
заполнение пропусков и Ordinal-кодирование
# SimpleImputer + OE
ord_pipe = Pipeline(
    [('simpleImputer_before_ord', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ord',  OrdinalEncoder(
                categories=[
                    ['junior', 'middle', 'sinior'],
                    ['low', 'medium', 'high'],
                ],
            )
        ),
```

```
    ]
)

#Создаём пайплайн для подготовки признаков
data_preprocessor = ColumnTransformer(
    [
        ('ohe', ohe_pipe, ohe_col),
        ('ord', ord_pipe, ord_col),
        ('num', MinMaxScaler(), num_col)
    ],
    remainder='passthrough'
)

#Финашльный пайплайн
pipe_final = Pipeline([
    ('preprocessor', data_preprocessor),
    ('models', DecisionTreeRegressor(random_state=RANDOM_STATE))
])

#Параметры пайплайна
param_grid = [
    # словарь для модели DecisionTreeClassifier()
    {
        'models': [DecisionTreeRegressor(random_state=RANDOM_STATE)],
       'models__max_depth': range(2,20),
        'models__min_samples_leaf': range(1,20),
        'models__min_samples_split': range(2,20),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },

    # словарь для модели LinearRegression()
    {
        'models': [LinearRegression()],
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough']
    }
]

RandomizedSearch = RandomizedSearchCV(
    pipe_final,
    param_grid,
    random_state = RANDOM_STATE,
    cv=3,
    scoring=smape_score,
    n_jobs=-1
)

RandomizedSearch.fit(train_job.drop('job_satisfaction_rate',axis=1),
train_job['job_satisfaction_rate'])
```

```
print('Лучшая модель и её параметры:\n\n',
RandomizedSearch.best_estimator_)
print ('Метрика лучшей модели на кросс-валидации:',
RandomizedSearch.best_score_)

# проверьте работу модели на тестовой выборке
# рассчитайте прогноз на тестовых данных
y_test_pred =
RandomizedSearch.predict(test_features.sort_values(by='id'))
print(f'Метрика SMAPE на тестовой выборке:
{smape(test_job.sort_values(by="id")["job_satisfaction_rate"],
y_test_pred)}')
```

```
Лучшая модель и её параметры:

 Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('ohe',

Pipeline(steps=[('simpleImputer_ohe',

SimpleImputer(strategy='most_frequent')),

('ohe',

OneHotEncoder(drop='first',

handle_unknown='ignore',

sparse_output=False))]),
                                                   ['dept',
'last_year_promo',

'last_year_violations']),
                                                  ('ord',

Pipeline(steps=[('simpleImputer_befor...

SimpleImputer(strategy='most_frequent')),

('ord',

OrdinalEncoder(categories=[['junior',

'middle',

'sinior'],

['low',
```

```
'medium',

'high']))]),
                                                  ['level',
'workload']),
                                                 ('num',
'passthrough',

Index(['employment_years', 'supervisor_evaluation', 'salary'],
dtype='object'))])),
                ('models',
                 DecisionTreeRegressor(max_depth=16,
min_samples_leaf=2,
                                       min_samples_split=3,
random_state=42))])
Метрика лучшей модели на тренировочной выборке: -14.739430487732074
Метрика SMAPE на тестовой выборке: 14.04335545915442
```

Промежуточные итоги

Лучшая модель - это DecisionTreeRegressor, встроенный в конвейер обработки данных. Эта модель была настроена с использованием следующих параметров: max_depth=16, min_samples_leaf=2, min_samples_split=3 и random_state=42.

DecisionTreeRegressor могла справиться лучше по нескольким причинам:

Способность к моделированию нелинейных зависимостей: Деревья решений хорошо справляются с задачами, где отношения между признаками и целевой переменной являются нелинейными в отличии от линейной регрессии. Они могут захватывать сложные структуры данных без необходимости предварительного преобразования признаков.

Настройка параметров: Параметры, такие как max_depth, min_samples_leaf и min_samples_split, были настроены таким образом, чтобы модель была достаточно сложной для захвата закономерностей в данных, но при этом обучающиеся деревья решений могут создавать слишком сложные деревья, которые плохо обобщают данные, то есть переобучатсья. Также деревья решений могут быть нестабильными, поскольку небольшие изменения в данных могут привести к созданию совершенно другого дерева.

Загрузка данных

Перейдем к задаче прогнозирования оттока сотрудников.

```
train_quit = pd.read_csv('/datasets/train_quit.csv') #Тренировочная
выборка
test_target_quit = pd.read_csv('/datasets/test_target_quit.csv')
#Целевой признак тестовой выборки
```

Первичное ознакомление с данными

```
dfs = {'train_quit':train_quit,'test_target_quit':test_target_quit}

for df in dfs:
    print(df)
    display(dfs[df].head(),dfs[df].info())
```

```
train_quit
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   id                     4000 non-null   int64 
 1   dept                   4000 non-null   object
 2   level                  4000 non-null   object
 3   workload               4000 non-null   object
 4   employment_years       4000 non-null   int64 
 5   last_year_promo        4000 non-null   object
 6   last_year_violations   4000 non-null   object
 7   supervisor_evaluation  4000 non-null   int64 
 8   salary                 4000 non-null   int64 
 9   quit                   4000 non-null   object
dtypes: int64(4), object(6)
memory usage: 312.6+ KB

       id        dept   level workload  employment_years
last_year_promo  \
0  723290       sales  middle     high                 2
no
1  814010       sales  junior   medium                 2
no
2  155091  purchasing  middle   medium                 5
no
3  257132       sales  junior   medium                 2
no
4  910140   marketing  junior   medium                 2
no

  last_year_violations  supervisor_evaluation  salary quit
0                   no                     4   54000   no
1                   no                     4   27600   no
2                   no                     1   37200   no
3                  yes                     3   24000  yes
4                   no                     5   25200   no

None

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

```
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      2000 non-null   int64 
 1   quit    2000 non-null   object
dtypes: int64(1), object(1)
memory usage: 31.4+ KB
```

```
       id quit
0  999029  yes
1  372846   no
2  726767   no
3  490105   no
4  416898  yes
```

```
None
```

Пропусков нет, это приятно.

Предобработка данных

```
for df in dfs:
    print(df,dfs[df].duplicated().sum())
```

```
train_quit 0
test_target_quit 0
```

Нет явных дубликатов.

```
for df in dfs:
    for column in dfs[df]:
        if dfs[df][column].dtype==object:
            display(dfs[df][column].unique())
```

```
array(['sales', 'purchasing', 'marketing', 'technology', 'hr'],
      dtype=object)
```

```
array(['middle', 'junior', 'sinior'], dtype=object)
```

```
array(['high', 'medium', 'low'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['no', 'yes'], dtype=object)
```

```
array(['yes', 'no'], dtype=object)
```

Не явных не видно.

Исследовательский анализ данных

Для начала будем действовать как в прошлой задаче.

Статистический анализ признаков

```
for df in dfs:
    for column in dfs[df]:
        if dfs[df][column].dtype!=object and column!='id':
            display(df ,column, dfs[df][column].describe())
```

```
'train_quit'
```

```
'employment_years'
```

```
count    4000.000000
mean        3.701500
std         2.541852
min         1.000000
25%         2.000000
50%         3.000000
75%         6.000000
max        10.000000
Name: employment_years, dtype: float64
```

```
'train_quit'
```

```
'supervisor_evaluation'
```

```
count    4000.000000
mean        3.474750
std         1.004049
min         1.000000
25%         3.000000
50%         4.000000
75%         4.000000
max         5.000000
Name: supervisor_evaluation, dtype: float64
```

```
'train_quit'
```

```
'salary'
```

```
count     4000.000000
mean     33805.800000
std      15152.415163
min      12000.000000
25%      22800.000000
50%      30000.000000
75%      43200.000000
max      96000.000000
Name: salary, dtype: float64
```

```
for df in dfs:
    for column in dfs[df]:
        if dfs[df][column].dtype==object:
            display(column, dfs[df][column].value_counts())
```

```
'dept'
```

```
sales         1438
technology     928
purchasing     588
marketing      582
hr             464
Name: dept, dtype: int64
```

```
'level'
```

```
junior    1949
middle    1694
sinior     357
Name: level, dtype: int64
```

```
'workload'
```

```
medium    2118
low       1208
high       674
Name: workload, dtype: int64
```

```
'last_year_promo'
```

```
no     3887
yes     113
Name: last_year_promo, dtype: int64
```

```
'last_year_violations'
```

```
no     3455
yes     545
Name: last_year_violations, dtype: int64
```

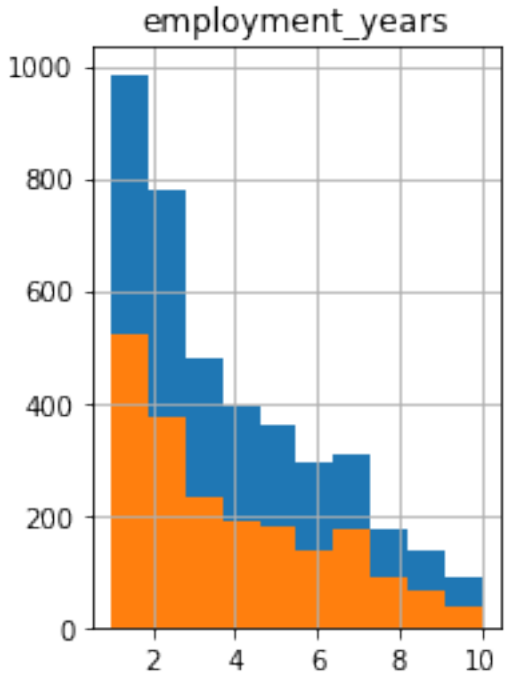
```
'quit'
```

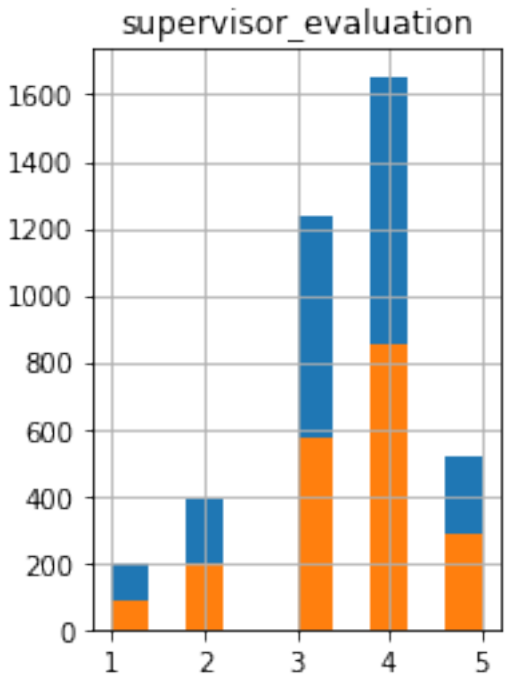
```
no     2872
yes    1128
Name: quit, dtype: int64
```

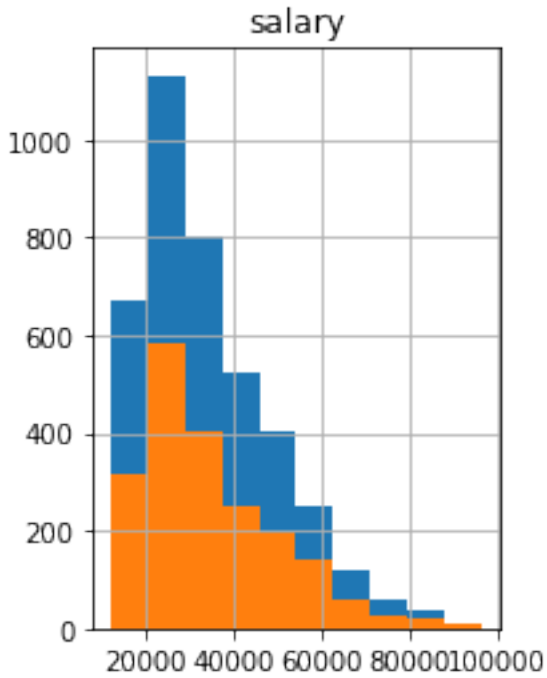
```
'quit'
```

```
no     1436
yes     564
Name: quit, dtype: int64
```

```
for column in test_features:
    if test_features[column].dtype!=object and column!='id':
        plt.subplot(1, 2, 1)
        dfs['train_quit'][column].hist()
        plt.title(column)
        plt.subplot(1, 2, 1)
        test_features[column].hist()
        plt.title(column)
        plt.show();
```







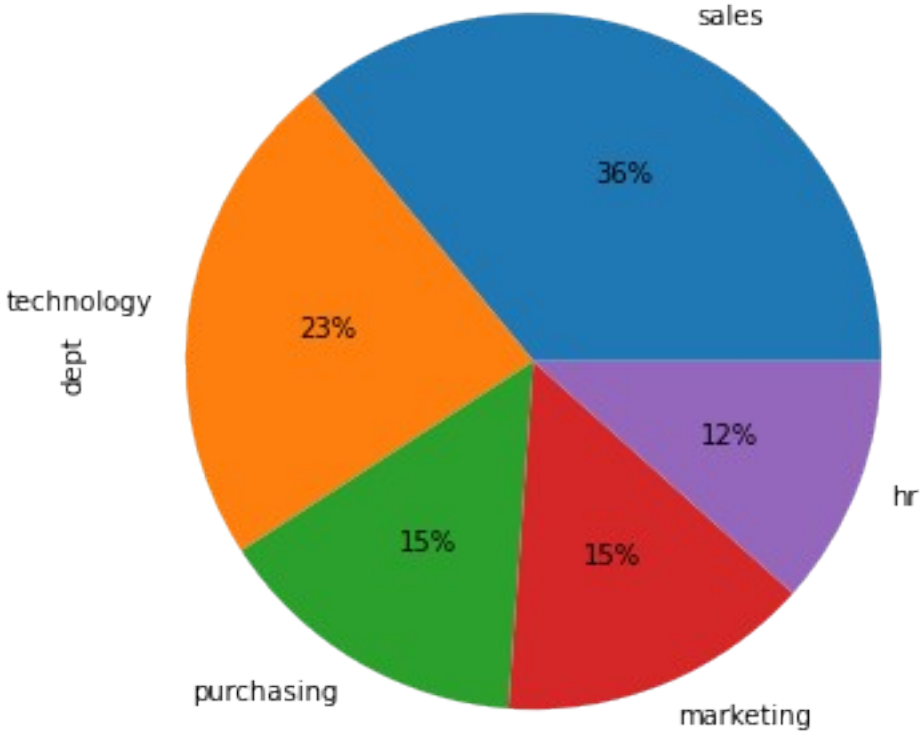

```
dfs['train_quit']
```

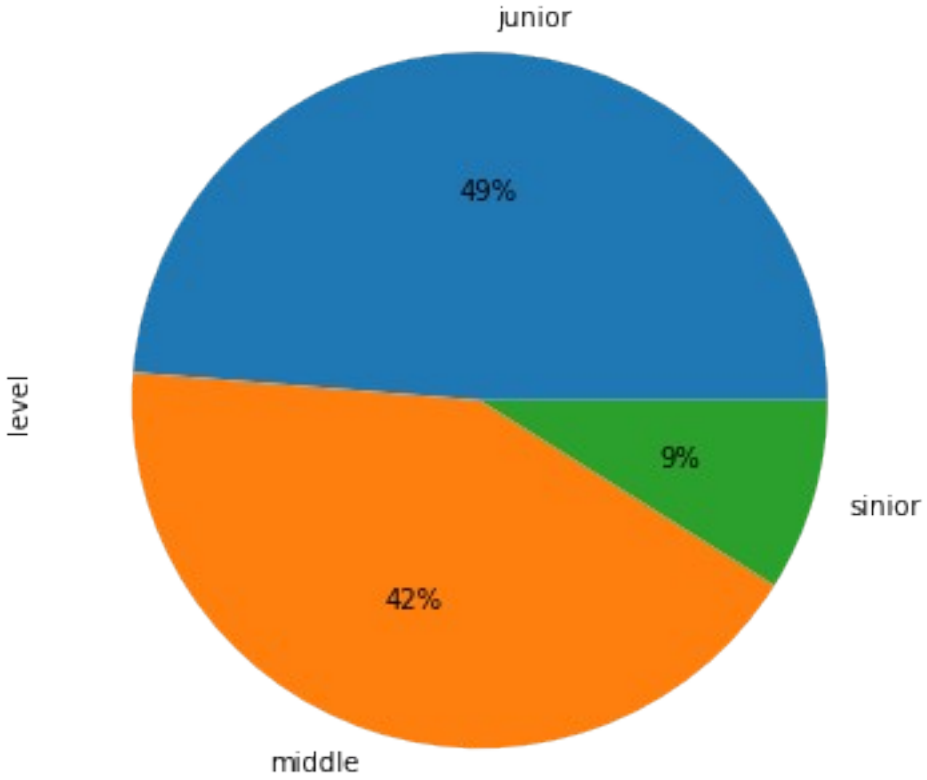
```
          id        dept   level workload  employment_years
last_year_promo  \
0     723290       sales  middle     high                 2
no
1     814010       sales  junior   medium                 2
```

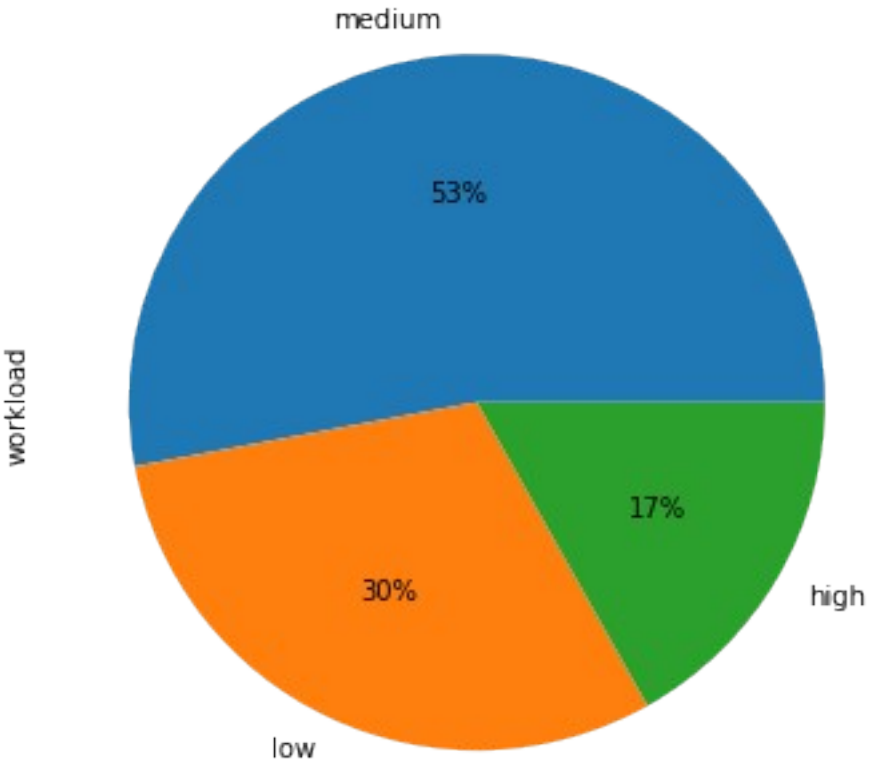
```
no
2     155091  purchasing  middle   medium                 5
no
3     257132       sales  junior   medium                 2
no
4     910140   marketing  junior   medium                 2
no
...      ...         ...     ...      ...               ...
...
3995  588809       sales  junior   medium                 4
no
3996  672059       sales  middle     high                 9
no
3997  536432  purchasing  junior      low                 2
no
3998  692133  purchasing  middle   medium                 2
no
3999  853842       sales  junior   medium                 2
no
```

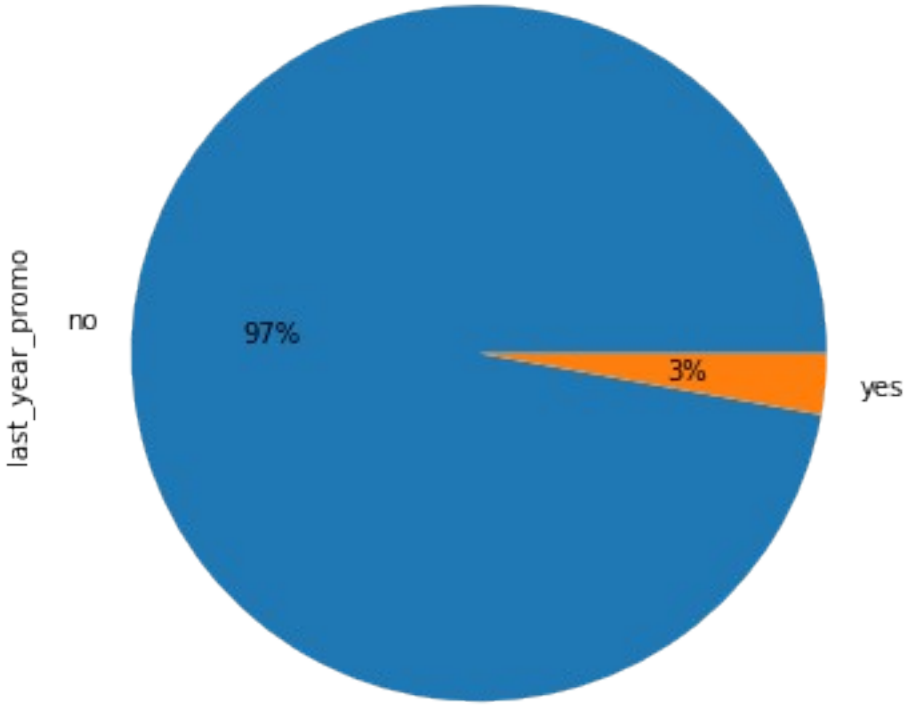
```
     last_year_violations  supervisor_evaluation  salary quit
0                      no                      4   54000   no
1                      no                      4   27600   no
2                      no                      1   37200   no
3                     yes                      3   24000  yes
4                      no                      5   25200   no
...                   ...                    ...     ...  ...
3995                   no                      3   26400   no
3996                   no                      4   52800   no
3997                  yes                      4   12000  yes
3998                   no                      4   33600   no
3999                   no                      3   27600  yes

[4000 rows x 10 columns]
```

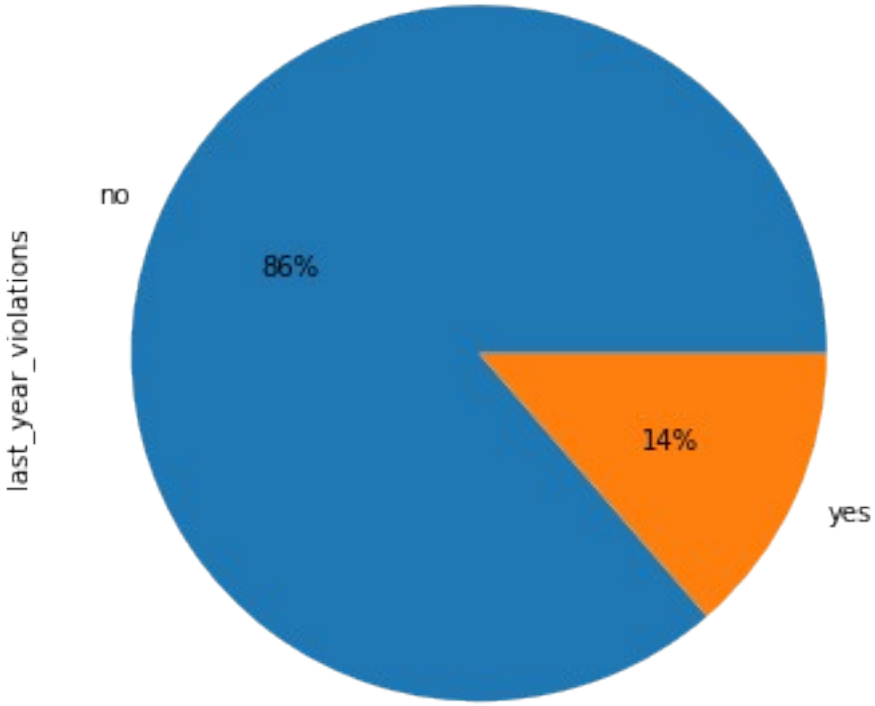
```
for column in dfs['train_quit']:
    if dfs['train_quit'][column].dtype==object:
        dfs['train_quit']
[column].value_counts().plot.pie(figsize=(6,6),autopct='%1.f%%')
        plt.show();
```

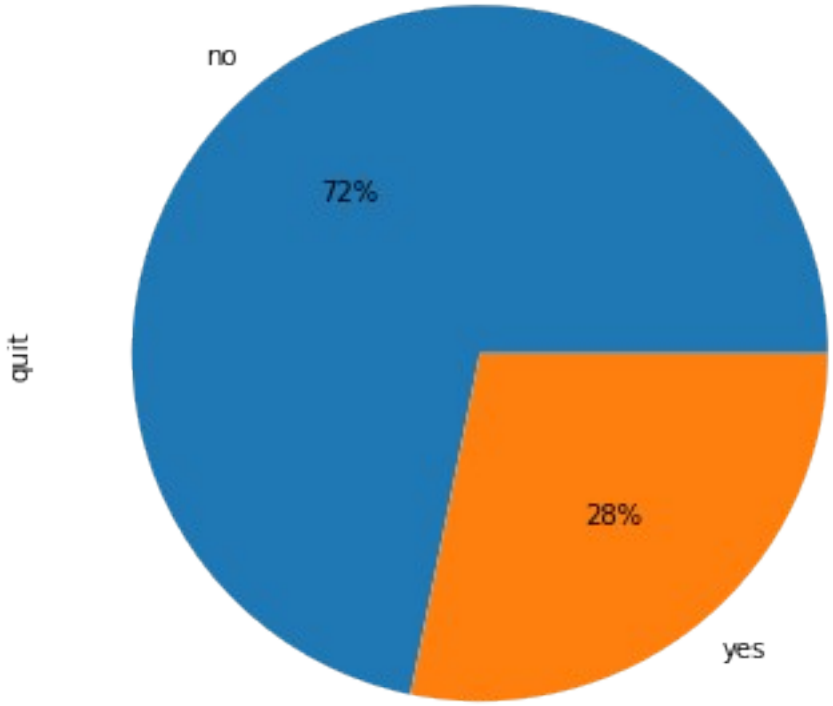

sales
36%
technology
dept
23%
12%
hr
15%
15%
purchasing
marketing


junior
49%
level
9%
sinior
42%
middle


medium
53%
workload
17%
30%
high
low


last_year_promo
no
97%
3%
yes


last_year_violations
no
86%
14%
yes



Значения имеют распределения аналогичные данным для предыдущей модели, но теперь то мы знаем, что четверть всех сотрудников уволится.

Корреляционный анализ

```
dfs['train_quit'].drop('id',axis=1).corr(method =
'spearman').style.background_gradient(cmap='coolwarm')
```

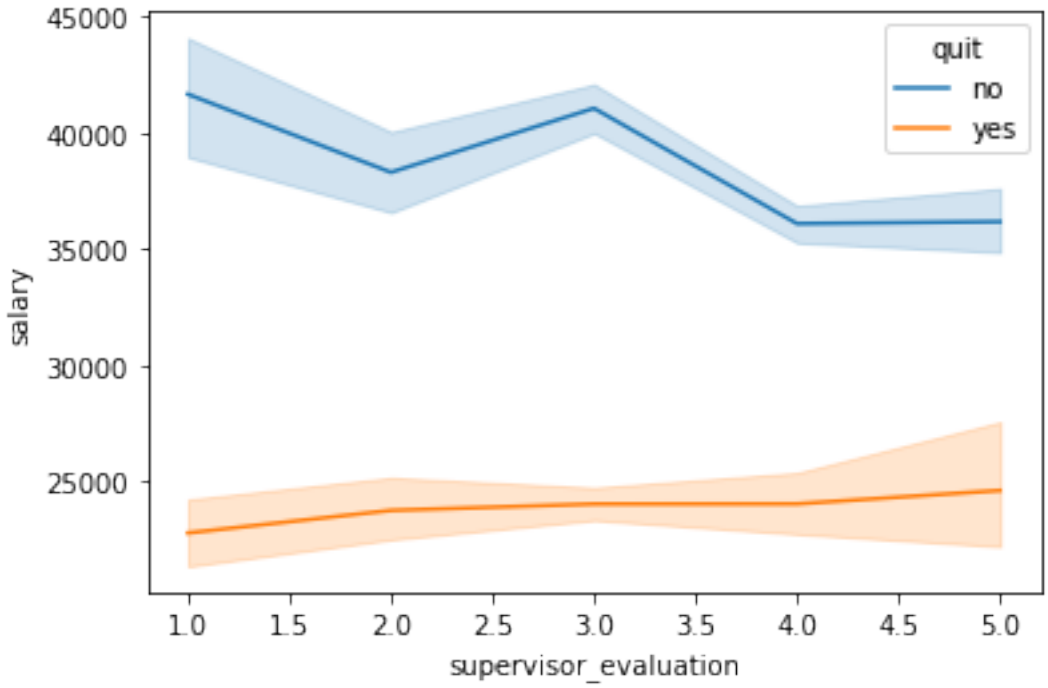
```
<pandas.io.formats.style.Styler at 0x7f54db105940>
```

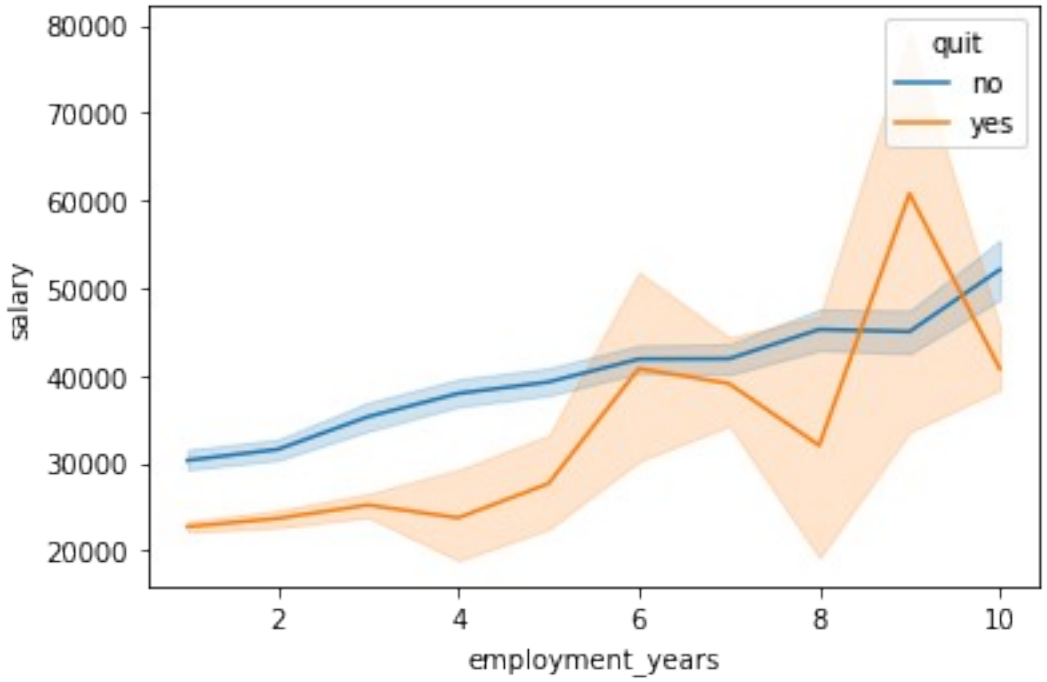
Те же данные(почти), те же корреляции.

Портрет уволившегося сотрудника

Для составления портрета уволившегося сотрудника, мы используем анализ доступных данных о сотрудниках. Посмотрим все что имеется в графическом виде.

```
sns.lineplot(data=train_quit, y='salary',x='supervisor_evaluation',
hue='quit')
plt.show();
sns.lineplot(data=train_quit, y='salary',x='employment_years',
hue='quit')
plt.show();
```





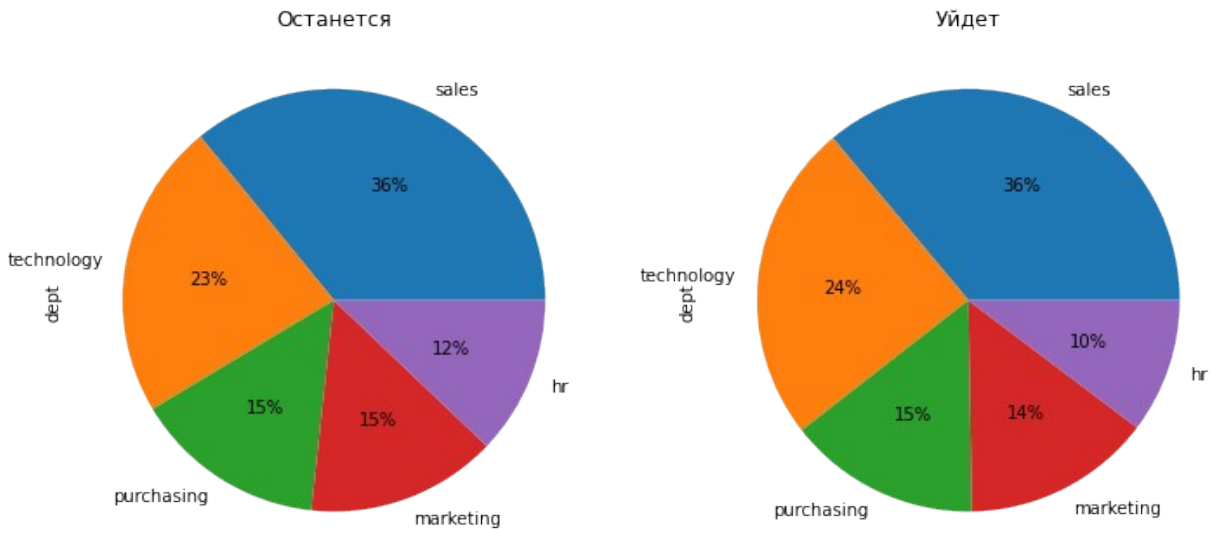
Сотрудники, склонные к увольнению, часто имеют низкую зарплату. Однако, даже сотрудники с большим стажем и высокой зарплатой могут рассматривать возможность смены работы. Это может быть связано с различными факторами, такими как:

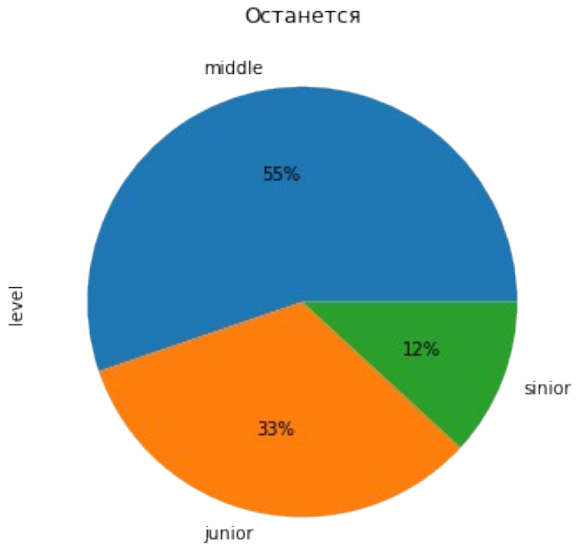
Недостаток карьерного роста: Сотрудники, которые не видят перспектив развития в текущей компании, могут искать новые возможности.

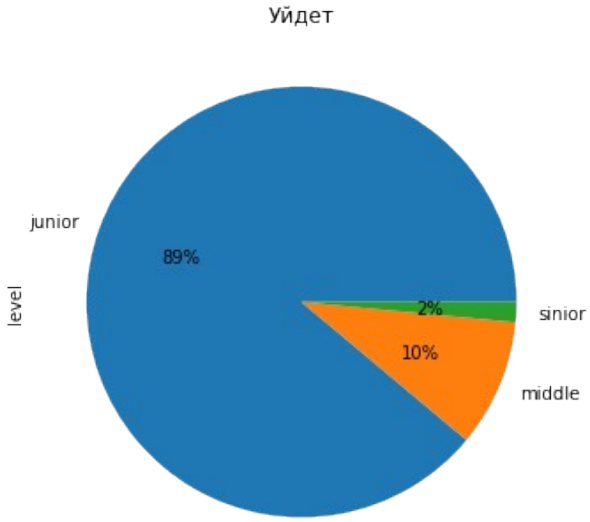
Неудовлетворенность рабочей средой: Конфликты на работе, недостаточная поддержка со стороны руководства или негативная корпоративная культура могут побудить сотрудников уйти.

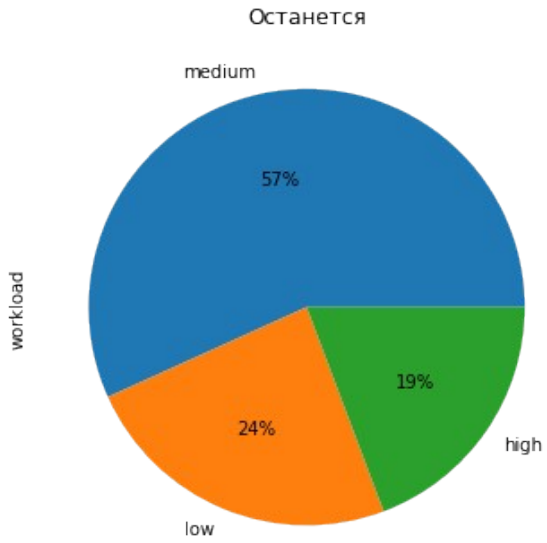
Поиск баланса между работой и личной жизнью: Сотрудники, стремящиеся к лучшему сочетанию работы и личной жизни, могут уволиться, если их текущая роль требует чрезмерных жертв.

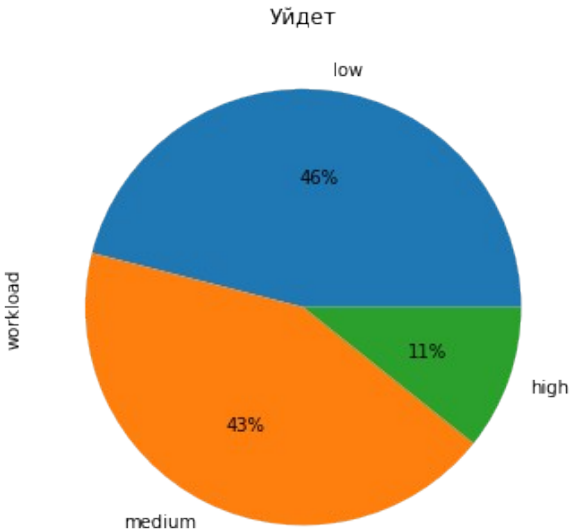
```
for column in train_quit:
    if train_quit[column].dtype==object and column!='quit':
        plt.subplot(1, 2, 1)
        train_quit[train_quit['quit']=='no']
[column].value_counts().plot.pie(figsize=(12,12),autopct='%1.f%%')
        plt.title(label='Останется')
        plt.subplot(1, 2, 2)
        train_quit[train_quit['quit']=='yes']
[column].value_counts().plot.pie(figsize=(12,12),autopct='%1.f%%')
        plt.title(label='Уйдет')
        plt.show();
```

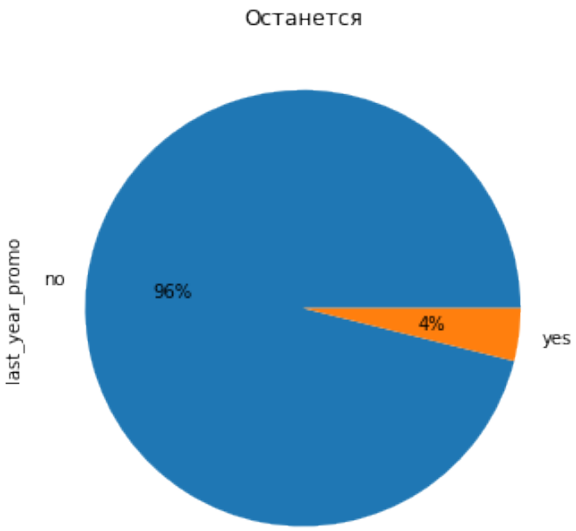


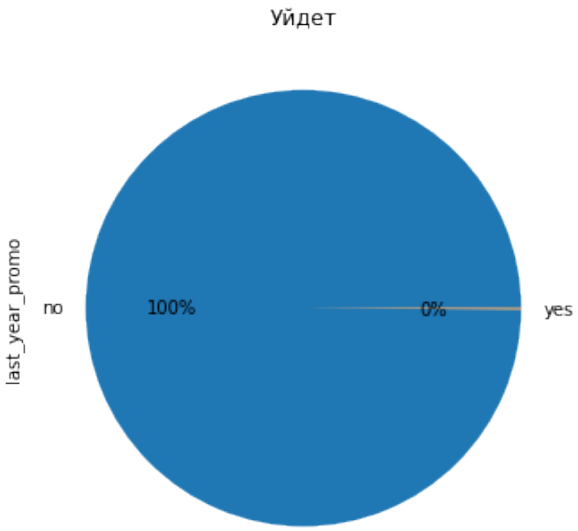

Останется
level
middle
55%
12%
sinior
33%
junior

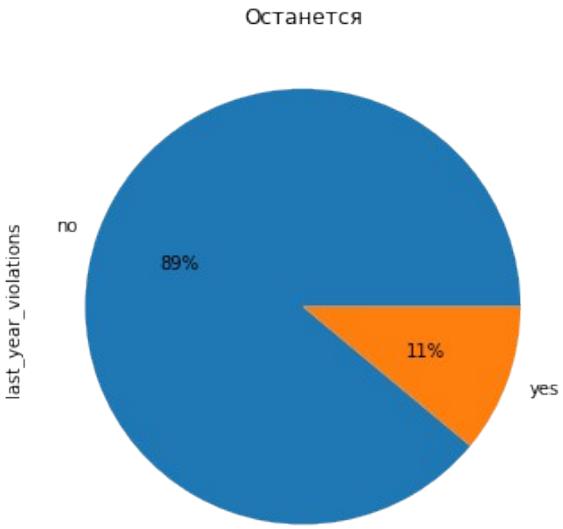

Уйдет
level
junior
89%
2%
sinior
10%
middle

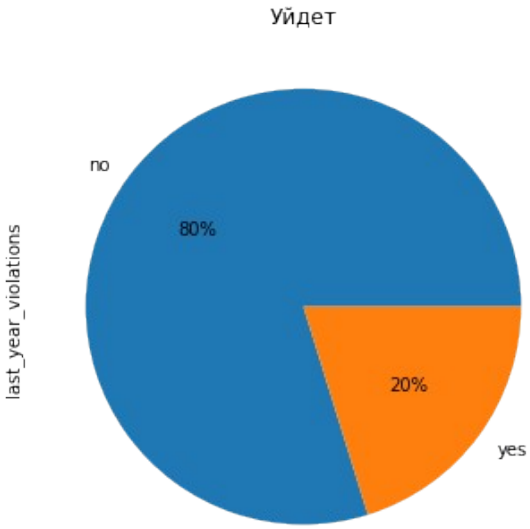

Останется
workload
medium
57%
19%
24%
high
low


Уйдет
workload
low
46%
11%
high
43%
medium



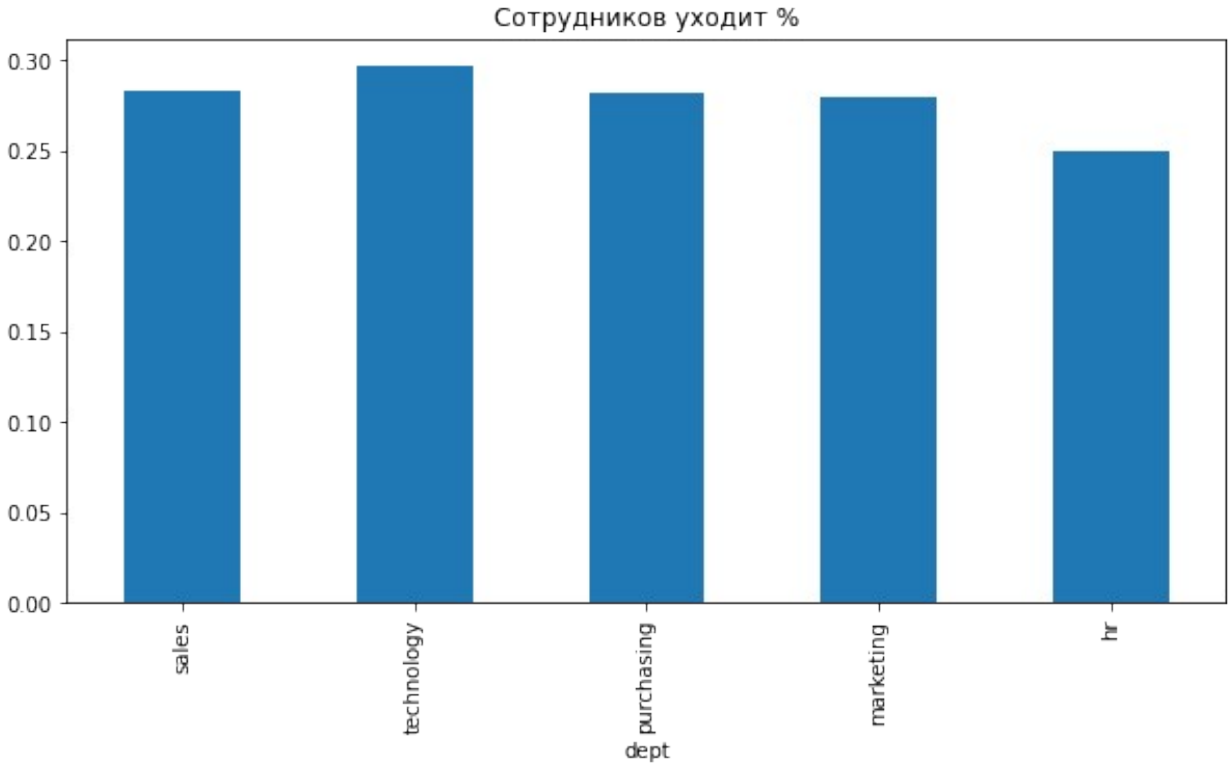


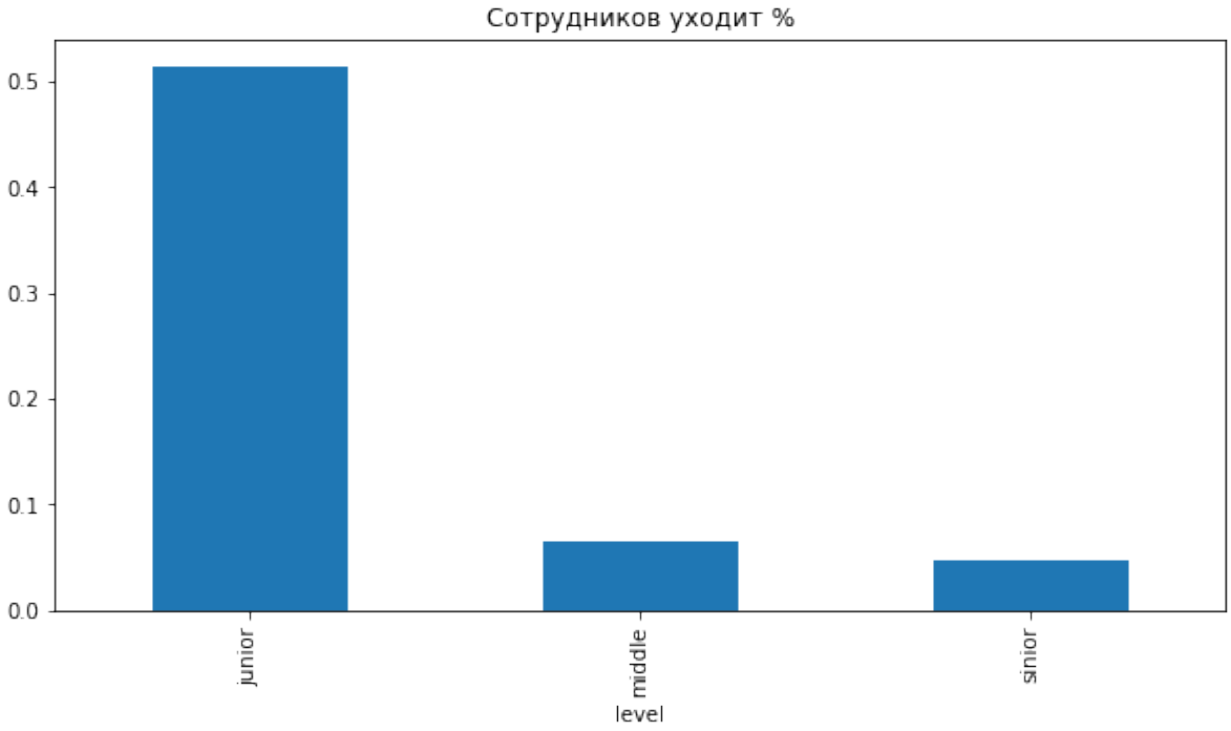


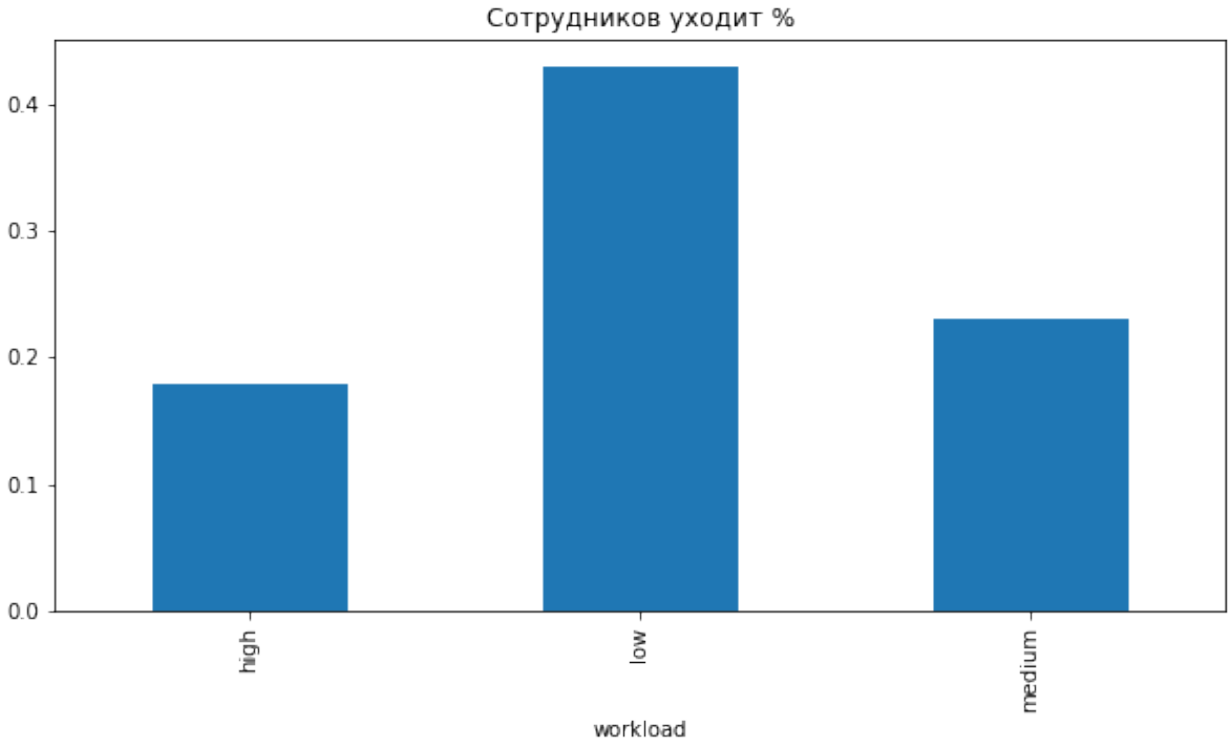


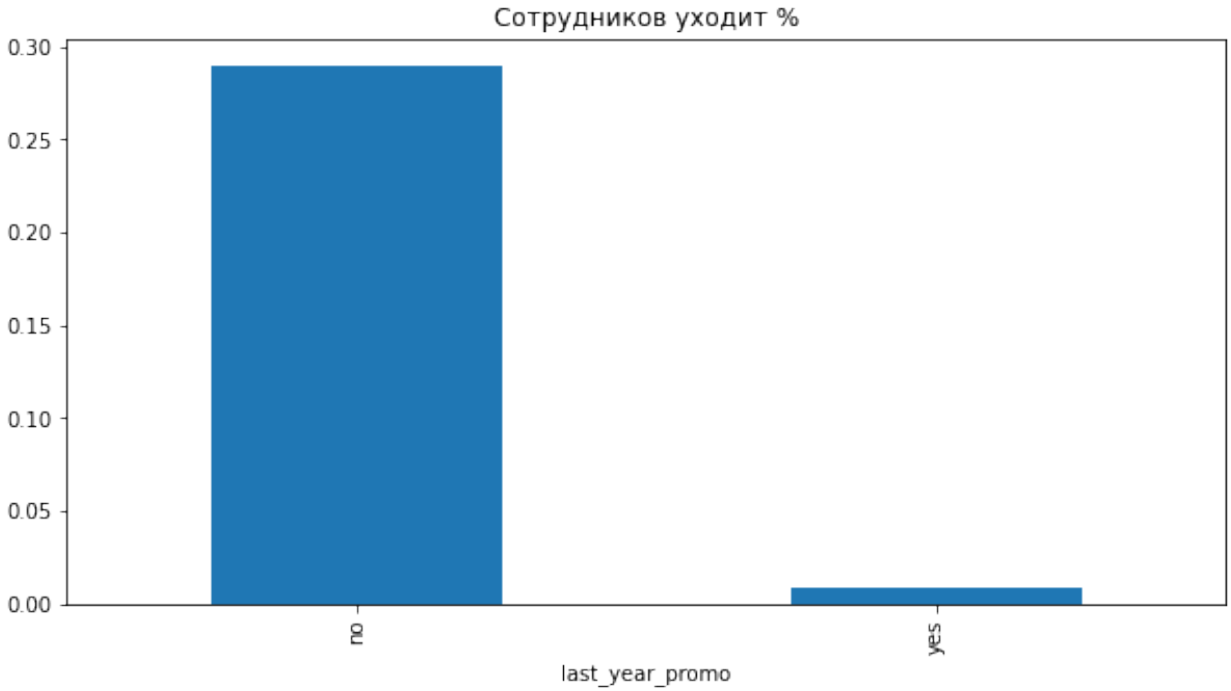
Большая часть ушедших работников принадлежит к крупнейшим отделам, неудивительно. Необходим анализ относительно количества ушедших к общему количеству для каждой группы.

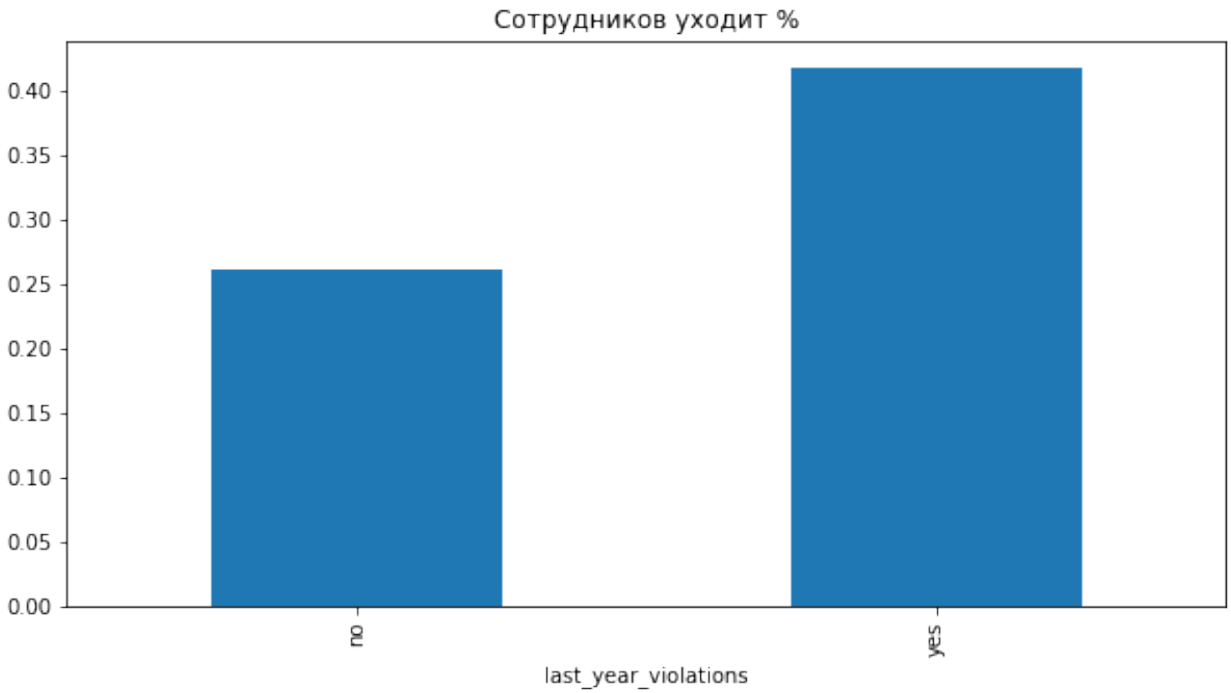
```
for column in train_quit:
    if train_quit[column].dtype==object and column!='quit':
        (train_quit[train_quit['quit']=='yes']
[column].value_counts()/train_quit[column].value_counts()).plot.bar(fi
gsize=(10,5))
        plt.xlabel(column)
        plt.title(label='Сотрудников уходит %')

        plt.show();
```


Сотрудников уходит %
0.30
0.25
0.20
0.15
0.10
0.05
0.00
sales
technology
purchasing
marketing
hr
dept


Сотрудников уходит %
0.5
0.4
0.3
0.2
0.1
0.0
junior
middle
sinior
level


Сотрудников уходит %
0.4
0.3
0.2
0.1
0.0
high
low
medium
workload


Сотрудников уходит %
0.30
0.25
0.20
0.15
0.10
0.05
0.00
no
yes
last_year_promo



Проанализировав даннын о сотрудниках можно сделать следующие выводы:

Отдел технологий имеет высокий уровень увольнений, что может указывать на проблемы внутри отдела или высокую конкуренцию на рынке труда в этой сфере.

Отдел HR имеет более низкий уровень увольнений, что может быть связано с лучшим пониманием ценности стабильности и карьерного роста внутри компании.

Сотрудники младших позиций увольняются чаще, что может быть связано с меньшей привязанностью к компании и большими возможностями для перехода на другие места.

Нагрузка на работе влияет на решение об увольнении, но не всегда очевидным образом. Сотрудники с низкой нагрузкой могут чувствовать себя недооцененными или скучать, в то время как высокая нагрузка может быть признаком вовлеченности и значимости работы.

Отсутствие повышения в течение года может сигнализировать сотрудникам о необходимости искать новые возможности для развития.

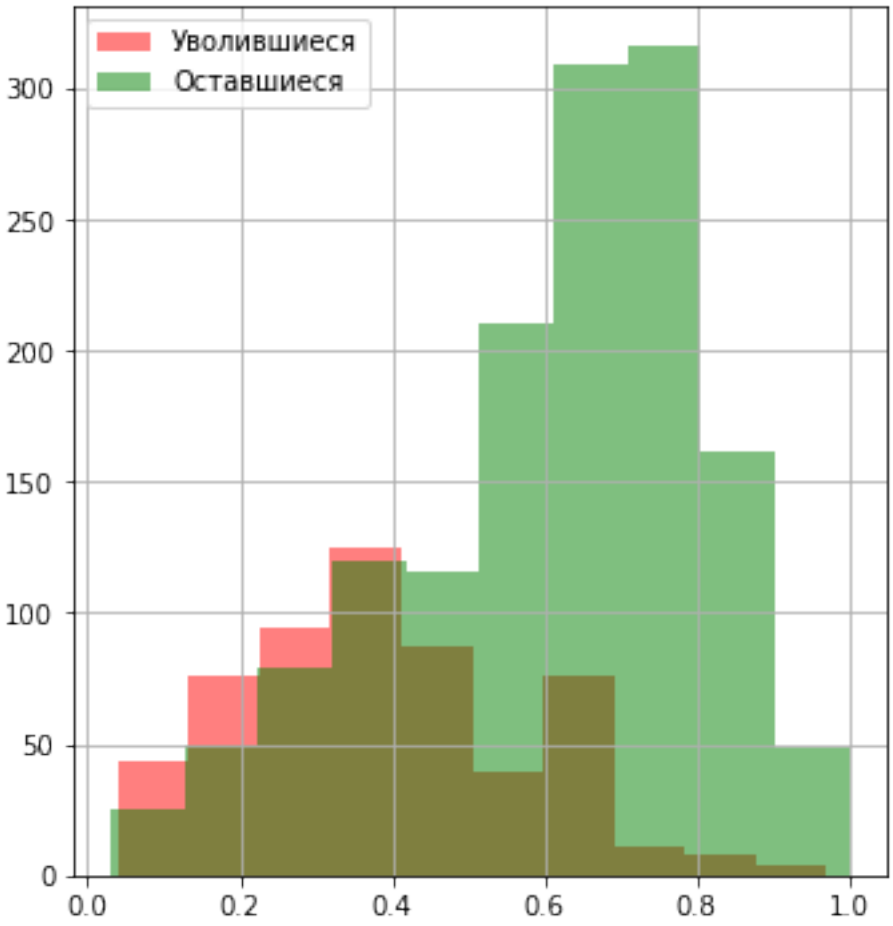
Сотрудники с нарушениями увольняются чаще, что может быть связано с низким уровнем удовлетворенности работой или проблемами в поведении.

Низкая зарплата является важным фактором, который может усиливать другие негативные аспекты работы.

Уровень удовлетворенности и увольнение

Уровень удовлетворённости работой является одним из ключевых показателей, который может влиять на решение сотрудника об увольнении.

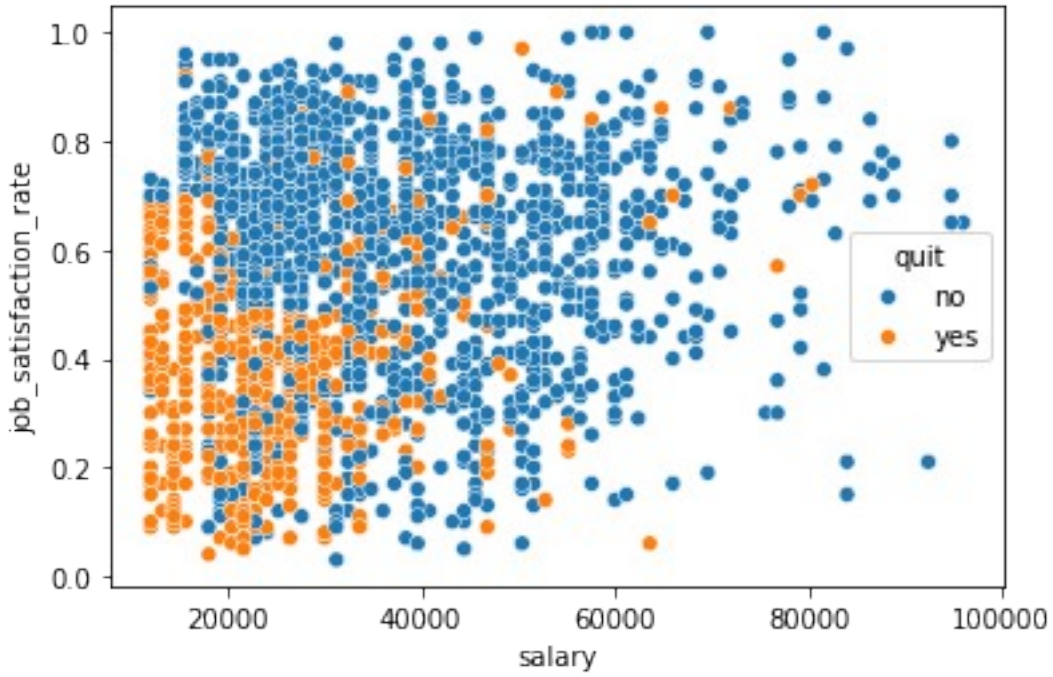
```
test = pd.merge(test_features, test_target_quit,
on='id').sort_values(by='id')

test = pd.merge(test, test_job, on='id')

plt.subplot(1, 2, 1)
test[test['quit']=='yes']['job_satisfaction_rate'].hist(alpha=0.5,
label = 'Уволившиеся',color='red',figsize=(12,6))
plt.subplot(1, 2, 1)
test[test['quit']=='no']['job_satisfaction_rate'].hist(alpha=0.5,
label = 'Оставшиеся',color='green',figsize=(12,6));
plt.legend();
```

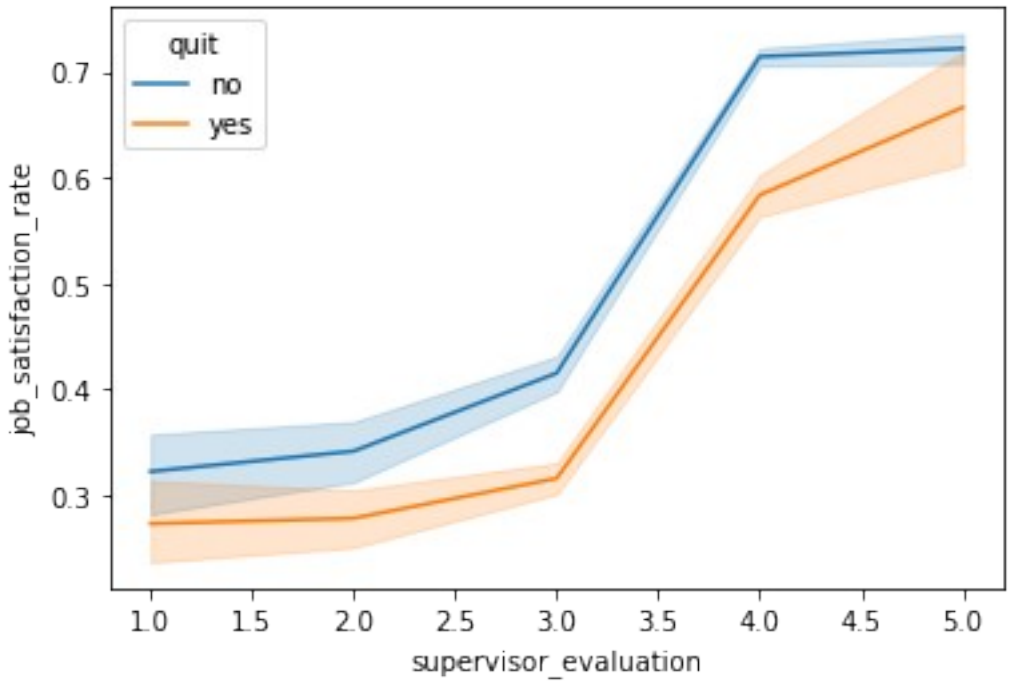


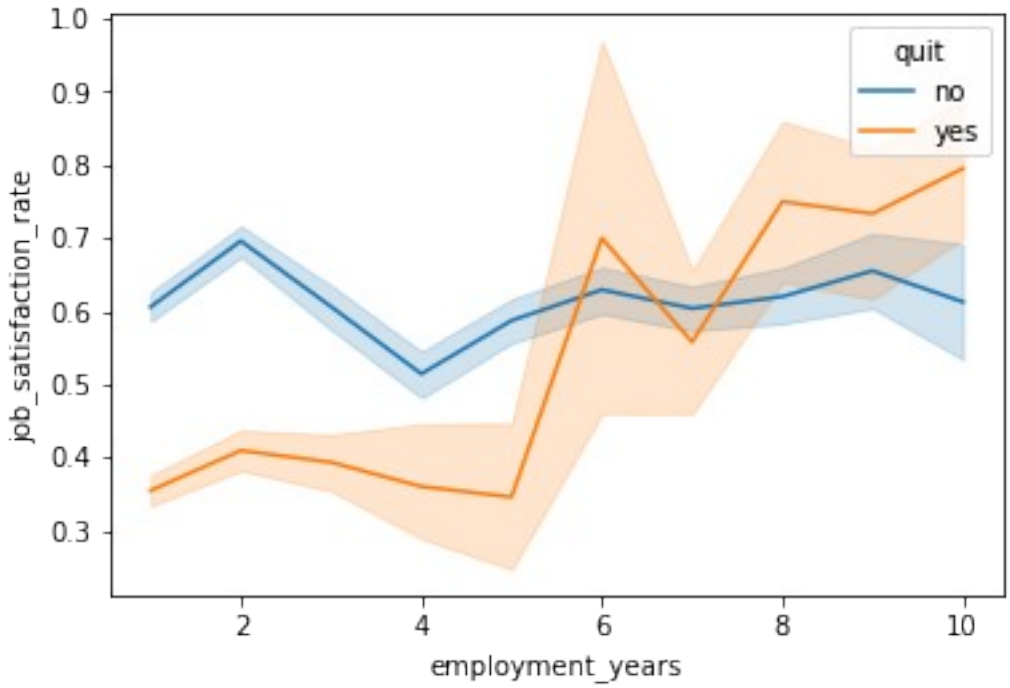
Средний показатель удовлетворенности работой среди уволившихся сотрудников ниже, чем показатель удовлетворенности работой у оставшихся.

```
sns.scatterplot(data=test, y='job_satisfaction_rate',x='salary',
hue='quit')
plt.show()
sns.lineplot(data=test,
y='job_satisfaction_rate',x='supervisor_evaluation', hue='quit')
```

```
plt.show();
sns.lineplot(data=test,
y='job_satisfaction_rate',x='employment_years', hue='quit')
plt.show();
```







Уровень удовлетворённости работой действительно играет значительную роль в решении сотрудника об увольнении. Обычно, сотрудники с более низким уровнем удовлетворённости склонны к поиску новых возможностей. Сотрудники удовлетворенные своей работой в компании и имеющие значительный опыт могут также задумывать о сменен места работы. Это может быть связано с желанием новых вызовов, стремлением к лучшему балансу между работой и личной жизнью, или же с поиском более высокой заработной платы и лучших условий труда.

Добавление нового входного признака

Попробуем добавить новый признак предсказанный предыдущей моделью к имеющимся данным.

```
dfs['train_quit']['job_satisfaction_rate'] =
RandomizedSearch.predict(dfs['train_quit'])
```

Вот так просто.

Подготовка данных и создание моделей

Без признака

```
RANDOM_STATE = 42
TEST_SIZE = 0.25

#Создаем списки признаков
ohe_col = ['dept', 'last_year_promo','last_year_violations']
```

```
num_col =
dfs['train_quit'].select_dtypes(exclude=['object']).columns.drop(['id'
,'job_satisfaction_rate'])
ord_col = ['level','workload']

ohe_pipe = Pipeline(
    [('simpleImputer_ohe', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ohe', OneHotEncoder(drop='first', handle_unknown='ignore',
sparse_output=False))
    ]
    )

# создаём пайплайн для подготовки признаков из списка ord_columns:
заполнение пропусков и Ordinal-кодирование
# SimpleImputer + OE
ord_pipe = Pipeline(
    [('simpleImputer_before_ord', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ord',  OrdinalEncoder(
                categories=[
                    ['junior', 'middle', 'sinior'],
                    ['low', 'medium', 'high'],
                ],
            )
        ),
    ]
)

#Создаём пайплайн для подготовки признаков
data_preprocessor = ColumnTransformer(
    [
        ('ohe', ohe_pipe, ohe_col),
        ('ord', ord_pipe, ord_col),
        ('num', MinMaxScaler(), num_col)
    ],
    remainder='passthrough'
)


#Финашльный пайплайн
pipe_final = Pipeline([
    ('preprocessor', data_preprocessor),
    ('models', DecisionTreeClassifier(random_state=RANDOM_STATE))
])

#Параметры пайплайна
param_grid = [

        # словарь для модели DecisionTreeClassifier()
```

```
    {
        'models': [DecisionTreeClassifier(random_state=RANDOM_STATE)],
       'models__max_depth': range(2,10),
        'models__min_samples_leaf': range(1,10),
        'models__min_samples_split': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },


    # словарь для модели KNeighborsClassifier()
    {
        'models': [KNeighborsClassifier()],
        'models__n_neighbors': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },

    # словарь для модели LogisticRegression()
    {
        'models': [LogisticRegression(
            random_state=RANDOM_STATE,
            solver='liblinear',
        )],
        'models__C': range(1,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },


    # Словарь для модели SVC
    {
        'models': [SVC(probability=True)],
        'models__kernel': ['poly','rbf','sigmoid'],
        'models__degree': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    }
]


RandomizedSearch = RandomizedSearchCV(
    pipe_final,
    param_grid,
    random_state = RANDOM_STATE,
    cv=6,
    scoring='roc_auc',
    n_jobs=-1
)
RandomizedSearch.fit(dfs['train_quit'].drop(['id',
```

```
'quit','job_satisfaction_rate'],axis=1), dfs['train_quit']['quit'])

print('Лучшая модель и её параметры:\n\n',
RandomizedSearch.best_estimator_)
print ('Метрика лучшей модели на тренировочной выборке:',
RandomizedSearch.best_score_)

# проверьте работу модели на тестовой выборке
# рассчитайте прогноз на тестовых данных
y_test_pred = RandomizedSearch.predict_proba(test.drop(['id',
'quit','job_satisfaction_rate'],axis=1))
print(f'Метрика ROC AUC на тестовой выборке:
{roc_auc_score(test["quit"], y_test_pred[:,1])}')
```

```
Лучшая модель и её параметры:

 Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('ohe',

Pipeline(steps=[('simpleImputer_ohe',

SimpleImputer(strategy='most_frequent')),

('ohe',

OneHotEncoder(drop='first',

handle_unknown='ignore',

sparse_output=False))]),
                                                   ['dept',
'last_year_promo',

'last_year_violations']),
                                                  ('ord',

Pipeline(steps=[('simpleImputer_befor...

SimpleImputer(strategy='most_frequent')),

('ord',

OrdinalEncoder(categories=[['junior',

'middle',

'sinior'],

['low',
```

```
'medium',

'high']))]),
                                                    ['level',
'workload']),
                                                   ('num',
MinMaxScaler(),

Index(['employment_years', 'supervisor_evaluation', 'salary'],
dtype='object'))])),
                ('models',
                 DecisionTreeClassifier(max_depth=8,
min_samples_leaf=7,
                                        min_samples_split=8,
                                        random_state=42))])
Метрика лучшей модели на тренировочной выборке: 0.9262497184332549
Метрика ROC AUC на тестовой выборке: 0.9233310367648513
```

С признаком

```
RANDOM_STATE = 42
TEST_SIZE = 0.25

#Создаем списки признаков
ohe_col = ['dept', 'last_year_promo','last_year_violations']
num_col =
dfs['train_quit'].select_dtypes(exclude=['object']).columns.drop(['id'
])
ord_col = ['level','workload']

ohe_pipe = Pipeline(
    [('simpleImputer_ohe', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ohe', OneHotEncoder(drop='first', handle_unknown='ignore',
sparse_output=False))
    ]
    )

# создаём пайплайн для подготовки признаков из списка ord_columns:
заполнение пропусков и Ordinal-кодирование
# SimpleImputer + OE
ord_pipe = Pipeline(
    [('simpleImputer_before_ord', SimpleImputer(missing_values=np.nan,
strategy='most_frequent')),
     ('ord',  OrdinalEncoder(
                categories=[
                    ['junior', 'middle', 'sinior'],
                    ['low', 'medium', 'high'],
                ],
```

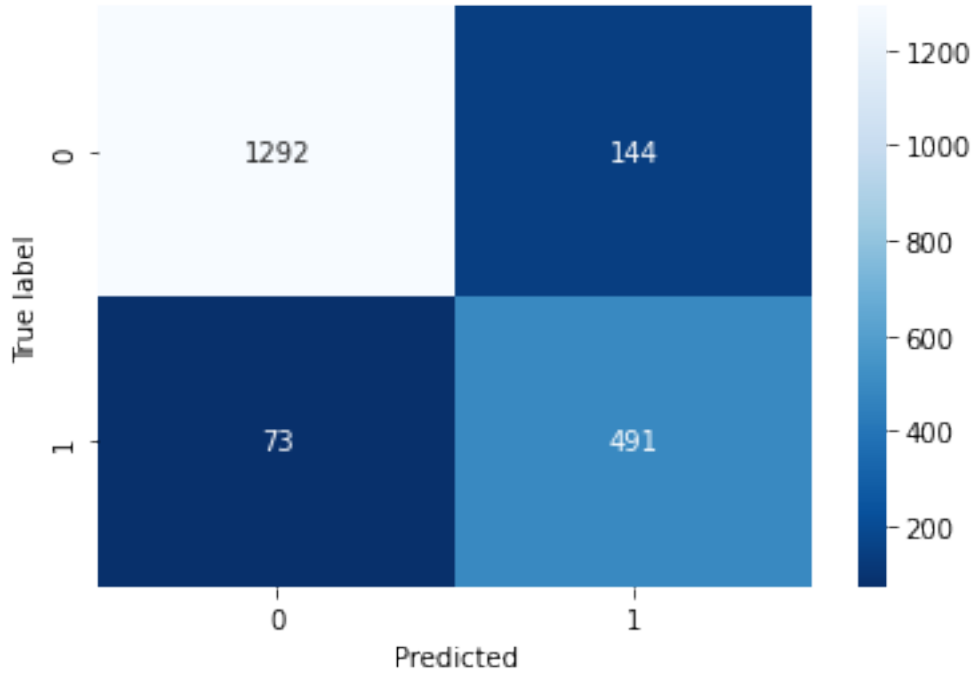
```
            )
        ),
    ]
)

#Создаём пайплайн для подготовки признаков
data_preprocessor = ColumnTransformer(
    [
        ('ohe', ohe_pipe, ohe_col),
        ('ord', ord_pipe, ord_col),
        ('num', MinMaxScaler(), num_col)
    ],
    remainder='passthrough'
)


#Финашльный пайплайн
pipe_final = Pipeline([
    ('preprocessor', data_preprocessor),
    ('models', DecisionTreeClassifier(random_state=RANDOM_STATE))
])

#Параметры пайплайна
param_grid = [

        # словарь для модели DecisionTreeClassifier()
    {
        'models': [DecisionTreeClassifier(random_state=RANDOM_STATE)],
       'models__max_depth': range(2,10),
        'models__min_samples_leaf': range(1,10),
        'models__min_samples_split': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },


    # словарь для модели KNeighborsClassifier()
    {
        'models': [KNeighborsClassifier()],
        'models__n_neighbors': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },

    # словарь для модели LogisticRegression()
    {
        'models': [LogisticRegression(
            random_state=RANDOM_STATE,
            solver='liblinear',
        )],
```

```
        'models__C': range(1,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    },


    # Словарь для модели SVC
    {
        'models': [SVC(probability=True)],
        'models__kernel': ['poly','rbf','sigmoid'],
        'models__degree': range(2,10),
        'preprocessor__num': [StandardScaler(), MinMaxScaler(),
'passthrough'],
    }
]


RandomizedSearch = RandomizedSearchCV(
    pipe_final,
    param_grid,
    random_state = RANDOM_STATE,
    cv=6,
    scoring='roc_auc',
    n_jobs=-1
)
RandomizedSearch.fit(dfs['train_quit'].drop(['id', 'quit'],axis=1),
dfs['train_quit']['quit'])

print('Лучшая модель и её параметры:\n\n',
RandomizedSearch.best_estimator_)
print ('Метрика лучшей модели на тренировочной выборке:',
RandomizedSearch.best_score_)

# проверьте работу модели на тестовой выборке
# рассчитайте прогноз на тестовых данных
y_test_pred = RandomizedSearch.predict_proba(test.drop(['id',
'quit'],axis=1))
print(f'Метрика ROC AUC на тестовой выборке:
{roc_auc_score(test["quit"], y_test_pred[:,1])}')
```

```
Лучшая модель и её параметры:

 Pipeline(steps=[('preprocessor',
                  ColumnTransformer(remainder='passthrough',
                                    transformers=[('ohe',

Pipeline(steps=[('simpleImputer_ohe',

SimpleImputer(strategy='most_frequent')),
```

```
('ohe',

OneHotEncoder(drop='first',

handle_unknown='ignore',

sparse_output=False))]),
                                                 ['dept',
'last_year_promo',

'last_year_violations']),
                                                ('ord',

Pipeline(steps=[('simpleImputer_befor...

SimpleImputer(strategy='most_frequent')),

('ord',

OrdinalEncoder(categories=[['junior',

'middle',

'sinior'],

['low',

'medium',

'high']]))]),
                                                 ['level',
'workload']),
                                                ('num',
'passthrough',

Index(['employment_years', 'supervisor_evaluation', 'salary',
       'job_satisfaction_rate'],
      dtype='object'))])),
                ('models',
                 DecisionTreeClassifier(max_depth=5,
min_samples_leaf=9,
                                        min_samples_split=8,
                                        random_state=42))])
Метрика лучшей модели на тренировочной выборке: 0.9287476514421761
Метрика ROC AUC на тестовой выборке: 0.9202164701001601

cm = confusion_matrix(test["quit"],
RandomizedSearch.predict(test.drop(['id', 'quit'],axis=1)))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues_r')
```

```
plt.ylabel('True label')
plt.xlabel('Predicted');
```



И снова деревья показали себя лучше других моделей, это произошло потому, что деревья решений часто хорошо справляются с нелинейными зависимостями и взаимодействиями между переменными, а размер дерева решений играет важную роль в его способности улавливать сложные закономерности в данных. Большое дерево с множеством узлов может лучше адаптироваться к обучающим данным, позволяя модели улавливать более тонкие и сложные взаимодействия между признаками. Однако это также может привести к переобучению, когда модель слишком точно подстраивается под обучающий набор данных и теряет способность обобщать на новых данных. Они могут автоматически выбирать важные переменные и игнорировать незначительные, что делает их очень эффективными для широкого спектра задач.

Введение спрогнозированого job satisfaction rate не сильно повлияло на прогноз модели, однако помогло существенно снизить размер дерева.

Общий вывод

В рамках проведенного исследования были созданы две модели: одна для предсказания увольнения сотрудников, а другая для оценки уровня их удовлетворенности работой. Анализ данных 4000 сотрудников и последующая проверка на выборке из 2000 человек позволили выявить ключевые факторы, влияющие на решение сотрудников об уходе из компании, а также на их удовлетворенность работой.

Исследовталеьский анализ позволили выявить различные тенденции в уровне удовлетворенности сотрудников компании.

Удовлетворенность работой: Оценка работы начальством оказалась значимым фактором удовлетворенности сотрудников, подчеркивая важность психологического комфорта и взаимоотношений в коллективе. Уровень зарплаты имел низкую корреляцию с удовлетворенностью, что указывает на приоритет рабочих условий над финансовым вознаграждением.

Факторы увольнения: Среди основных причин, способствующих желанию сотрудников искать новую работу, были выявлены низкая зарплата, отсутствие карьерного роста, негативные отношения в коллективе и состояние рынка труда.

Рекомендации для бизнеса: Для снижения оттока сотрудников рекомендуется сосредоточить усилия на улучшении внутреннего климата в коллективах, особенно в крупных отделах, играющих ключевую роль в деятельности компании. Кроме того, следует уделить внимание распределению рабочей нагрузки и разработать индивидуальные подходы к удержанию ключевых сотрудников а также сотрудников низшего звена при необходимости.

Это исследование подчеркивает, что для повышения удовлетворенности и снижения оттока сотрудников необходим комплексный подход, включающий как материальные, так и нематериальные аспекты трудовой деятельности. Внимание к деталям и индивидуальный подход к каждому сотруднику могут стать ключом к созданию продуктивной и лояльной команды.