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)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Тема: «Прогнозирование уклонения от уплаты налогов компаниями (MIPYME) на Кубе»

Слушатель Суарес Гомес Хорхе Алехандро

Москва, 2024

Содержание

Введение

MIPYME это микро , малое или среднее предприятие, которое осуществляет в стране свою деятельность в одном из следующих секторов: услуги, торговля, промышленность, сельское хозяйство, гастрономия и строительство.

Микропредприятия : обычно они состоят не более чем из 10 человек и, как правило, являются семейными предприятиями. Они созданы людьми, которые изначально не обладают очень большим капиталом.

Малое предприятие : в отличие от микропредприятия, малые предприятия могут насчитывать от 11 до 35 человек

Средний предприятие : эти типы компаний крупнее двух предыдущих, поскольку в них может работать от 36 до 100 человек,

В экономике, контролируемой государством на протяжении десятилетий, и в условиях американских санкций в отношении Кубы на протяжении более 60 лет. страна приняла меры по сокращению дефицита продовольствия, и с 2021 года по настоящее время зарегистрировано около 9900 таких предприятий, а по состоянию на конец января 2023 года, дату проведения этого исследования, было зарегистрировано 7787 малых и средних предприятий.

Цель этой работы - спрогнозировать, нарушают ли МСБ подоходный налог и выплаты рабочей силе, на основе данных каждого налогоплательщика. Согласно исследованию, проведенному Управлением налоговых платежей Кубы, только за первые 3 месяца 2024 года было совершено около 80 852 операций контроль и более 819 594 000 миллионов причитающихся кубинские песо, что эквивалентно почти 7 миллионам долларов, что определяет необходимость усиления контроля над этими экономическими субъектами.

Выявление нарушений на Кубе представляет собой сложный процесс, поскольку раньше не существовало контроля над этими типами компаний, поэтому, как и традиционно , необходимо проводить длительные и тщательные проверки для

каждого налогоплательщика, что ставит задачу прогнозирования решения. Снижение потерянного сбора налогов, что является часть государственный бюджет. Прогнозирование состоит в том , чтобы определить, платит ли налогоплательщик налоги или нет, на основе таких данных, как выплаты по доходам, выплаты работникам и численность сотрудников компании. В ходе исследовательской работы было разработано несколько моделей, способных прогнозировать с высокой вероятностью

1. Аналитическая часть

1.1. Постановка задачи

Для исследовательской работы было предоставлено 4 файла: info_contrib.xlsx (с данными налогоплательщиков и состоит из 2156 строки и 8 столбцов) и imp_ingreso.xlsl (данные о выплатах 10% доходов налогоплательщиков, состоящий из 2156 строки и 13 столбцов). imp_salario.xlsl (данные о выплате 5% заработной платы работникам и 12,5 процента социального обеспечения состоит из 2156 строки и 13 столбцов) и auditoria.xlsl (с данными о проведенном аудите этих налогоплательщиков и состоит из 2156 строки и 2 столбцов)

1. Загружаем исходные данные

```
[76]: #Загружаем датасет INFO
ruta1 = 'C:/Alejandro/info_contrib.xlsx'
info_contrib = pd.read_excel(ruta1)
info_contrib.shape
```

```
[76]: (2156, 8)
```

```
[77]: # 2 налог на продажу (imp_ingreso)
ruta2 = 'C:/Alejandro/imp_ingreso.xlsx'
imp_ingreso = pd.read_excel(ruta2)
imp_ingreso.shape
```

```
[77]: (2156, 13)
```

```
[78]: # 3 налог на рабочую силу и социальное обеспечение (imp_salario)
ruta3 = 'C:/Alejandro/imp_salario.xlsx'
imp_salario = pd.read_excel(ruta3)
imp_salario.shape
```

```
[78]: (2156, 13)
```

```
[79]: # 4 проверка Нарушение от уплаты налогов
ruta4 = 'C:/Alejandro/auditoria.xlsx'
auditoria = pd.read_excel(ruta4)
auditoria.shape
```

```
[79]: (2156, 2)
```

```
[80]: new_names = {'Idcontribuyente': 'id', 'Edad': 'Возраст', 'Sexo': 'Пол', 'MIPYME': 'Компания','Actividad': 'Деятельность', 'Inscripcion': 'Регистрация', '
info_contrib = info_contrib.rename(columns=new_names)
info_contrib.head()
```

[80]:

	id	Возраст	Пол	Компания	Деятельность	Регистрация	Сотрудники	Местонахождение
0	1	41	Masculino	Micro	Construccion	2022-03-12	5	Occidente
1	2	43	Masculino	Pequeña	Construccion	2022-05-08	30	Centro
2	3	47	Masculino	Pequeña	Gastronomicos	2022-04-05	31	Centro
3	4	36	Masculino	Micro	Construccion	2022-02-14	6	Occidente
4	5	42	Masculino	Pequeña	Gastronomicos	2021-11-11	22	Occidente

```
[81]: new_names = {'Idcontribuyente': 'id', 'Enero': 'Январь', 'Febrero': 'Февраль' , 'Marzo': 'Март', 'Abril': 'Апрель', 'Mayo': 'Май', 'Junio': 'Июнь', 'Juli
imp_ingreso = imp_ingreso.rename(columns=new_names)
imp_ingreso.head()
```

[81]:

	id	Январь	Февраль	Март	Апрель	Май	Июнь	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
0	1	228485.0	231583.0	227883.0	224055.0	218433.0	217756.0	220297	228083	226939	221586	230104	220983
1	2	1295622.0	1327598.0	1309699.0	1320146.0	1333394.0	1335997.0	1338167	1333049	1302617	1322652	1311463	1317768
2	3	1190168.0	1209205.0	1190435.0	1193072.0	1183155.0	1189882.0	1182446	1203122	1161748	1188487	1170637	1223327
3	4	275082.0	283708.0	276912.0	283177.0	285418.0	285704.0	286413	286244	270626	271593	272354	286293
4	5	1011603.0	1002413.0	974753.0	983996.0	1017701.0	1010577.0	1017118	995798	1012133	1008731	1012448	993869

Рисунок 1 и 2 - пример начала работы с файлом

1.2. Описание используемых методов

Данная задача в рамках классификации категорий машинного обучения относится к машинному обучению с учителем и традиционно является регрессионной задачей, Хорошо подходит для задач с четко определенными входными и выходными данными:

- KneighborsClassifier (К-ближайших соседей)
- DecisionTreeClassifier (Дерево решений)
- GaussianNB
- RandomForestClassifier (Случайный лес)
- LogisticRegression
- Support Vector Regression (метод опорных векторов SVR)

Метод ближайших соседей - К-ближайших соседей (KneighborsClassifier) ищет ближайшие объекты с известными значения целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами. Алгоритм находит расстояния между запросом и всеми примерами в данных, выбирая определенное количество примеров (k), наиболее близких к запросу, затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Достоинства метода: прост в реализации и понимании полученных результатов; имеет низкую чувствительность к выбросам; не требует построения модели; допускает настройку нескольких параметров; позволяет делать дополнительные допущения; универсален; находит лучшее решение из возможных; решает задачи небольшой размерности.

Недостатки метода: замедляется с ростом объёма данных; не создаёт правил; не обобщает предыдущий опыт; основывается на всем массиве доступных исторических данных; невозможно сказать, на каком основании строятся ответы; сложно выбрать близость метрики; имеет высокую зависимость результатов

классификации от выбранной метрики; полностью перебирает всю обучающую выборку при распознавании; имеет вычислительную трудоёмкость.

Дерево решений (DecisionTreeClassifier): Алгоритм разделяет данные на подмножества, основываясь на значении признаков. Для каждого раздела он выбирает признак, который наиболее эффективно разделяет данные, оптимизируя критерий чистоты (например, энтропию или Gini-индекс). Процесс повторяется рекурсивно для каждого подмножества, пока не будут достигнуты определенные условия (например, максимальная глубина дерева или минимальное количество образцов в узле).

Преимущества :Прост в понимании: Древовидная структура позволяет легко визуализировать и интерпретировать процесс принятия решения. Не требует нормализации данных: Алгоритм может работать с данными, имеющими различные масштабы. Устойчив к шуму: Алгоритм может хорошо справляться с данными, содержащими шум и выбросы. Может использоваться для задач feature engineering: Алгоритм может использоваться для создания новых признаков, основанных на комбинациях существующих.

Недостатки :Склонен к переобучению: Алгоритм может создавать слишком сложные деревья, которые хорошо работают на обучающих данных, но плохо обобщаются на новых данных. Нестабилен: Небольшие изменения в обучающих данных могут привести к значительному изменению структуры дерева. Может быть неэффективным для задач с высокой размерностью: Алгоритм может быть медленным для обработки данных с большим количеством признаков.

Примеры использования DecisionTreeClassifier: Классификация клиентов: Предсказание, является ли клиент лояльным или нет. Распознавание изображений: Классификация изображений по категориям (например, животных, автомобилей). Медицинская диагностика: Предсказание наличия заболевания.

GaussianNB (Gaussian Naive Bayes) - это алгоритм машинного обучения, который использует теорему Байеса для классификации данных. Он классифицирует данные на основе вероятности принадлежности объекта к определенному классу, учитывая значения его признаков. Алгоритм предполагает, что признаки распределены по нормальному (гауссовскому) закону.

Основные особенности GaussianNB: Простой: Алгоритм легко реализовать и обучить. Быстрый: Классификация новых данных происходит быстро. Наивный: Алгоритм предполагает, что признаки независимы друг от друга. Это не всегда справедливо, но часто работает хорошо. Непараметрический: Алгоритм не делает никаких предположений о распределении данных, кроме нормальности признаков.

Подходит для многомерных данных: Может использоваться для классификации данных с большим количеством признаков.

Преимущества: Прост в реализации и обучении. Работает быстро даже на больших наборах данных. Устойчив к шуму и пропущенным данным.

Недостатки: Предположение о независимости признаков может быть нереалистичным. Могут быть проблемы с обучением, если данные не распределены по нормальному закону.

Применение:

GaussianNB часто используется в следующих задачах:

Классификация текста

Фильтрация спама

Распознавание образов

Случайный лес (RandomForest) — это множество решающих деревьев. Универсальный алгоритм машинного обучения с учителем, представитель ансамблевых методов. Если точность дерева решений оказалось недостаточной, мы можем множество моделей собрать в коллектив.

Достоинства метода: не переобучается; не требует предобработки входных данных; эффективно обрабатывает пропущенные данные, данные с большим числом классов и признаков; имеет высокую точность предсказания и внутреннюю оценку обобщающей способности модели, а также высокую параллелизуемость и масштабируемость.

Недостатки метода: построение занимает много времени; сложно интерпретируемый; не обладает возможностью экстраполяции; может недо обучаться; трудоёмко прогнозируемый; иногда работает хуже, чем линейные методы.

Логистическая регрессия (Logistic Regression): это статистический метод машинного обучения, который используется для прогнозирования вероятности принадлежности объекта к определенной категории. Она основана на предположении, что зависимая переменная (целевая переменная) имеет бинарное распределение (т.е. две возможные категории).

Применение логистической регрессии: Классификация: Логистическая регрессия широко используется для решения задач классификации, например: Сортировка спама: Определение, является ли электронное письмо спамом или нет. Оценка кредитоспособности: Прогнозирование вероятности того, что заемщик вернет кредит. Диагностика болезней: Предсказание наличия или отсутствия определенного заболевания. Прогнозирование: Логистическая регрессия также может использоваться для прогнозирования вероятности события, например: Прогнозирование продаж: Предсказание количества продаж продукта в определенном регионе. Прогнозирование оттока клиентов: Оценка вероятности того, что клиент прекратит пользоваться услугами компании.

Преимущества логистической регрессии: Простота: Модель относительно легко понять и интерпретировать. Скорость: Тренировка и прогнозирование выполняются быстро. Эффективность: Модель часто дает хорошие результаты даже с небольшими объемами данных.

Недостатки логистической регрессии: Линейность: Модель предполагает линейную зависимость между предикторами и целевой переменной, что может ограничивать ее применение. Слабая производительность в случае сложных нелинейных взаимосвязей: Модель может не справиться с данными, имеющими сложные нелинейные зависимости. Проблемы с выбросами: Модель чувствительна к выбросам в данных.

Метод опорных векторов (Support Vector Regression) – этот бинарный линейный классификатор был выбран, потому что он хорошо работает на небольших датасетах. Данный алгоритм – это алгоритм обучения с учителем, использующихся для задач классификации и регрессионного анализа, это контролируемое обучение моделей с использование схожих алгоритмов для анализа данных и распознавания шаблонов. Учитывая обучающую выборку, где алгоритм помечает каждый объект, как принадлежащий к одной из двух категорий, строит модель, которая определяет новые наблюдения в одну из категорий.

Модель метода опорных векторов – отображение данных точками в пространстве, так что между наблюдениями отдельных категорий имеется разрыв, и он максимален.

Каждый объект данных представляется как вектор (точка) в p-мерном пространстве. Он создаёт линию или гиперплоскость, которая разделяет данные на классы.

Достоинства метода: для классификации достаточно небольшого набора данных. При правильной работе модели, построенной на тестовом множестве, вполне возможно применение данного метода на реальных данных. Эффективен при большом количестве гиперпараметров. Способен обрабатывать случаи, когда гиперпараметров больше, чем количество наблюдений. Существует возможность гибко настраивать разделяющую функцию. Алгоритм максимизирует

разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации.

Недостатки метода: неустойчивость к шуму, поэтому в работе была проведена тщательнейшая работа с выбросами, иначе в обучающих данных шумы становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости; для больших наборов данных требуется долгое время обучения; достаточно сложно подбирать полезные преобразования данных; параметры модели сложно интерпретировать, поэтому были рассмотрены и другие методы.

1.3. Разведочный анализ данных

(EDA - Exploratory Data Analysis) — это процесс визуализации и анализа данных с целью получить представление о структуре данных, выявить закономерности, аномалии и взаимосвязи между переменными. EDA — это ключевой этап в процессе анализа данных, который помогает понять данные, сформулировать гипотезы и выбрать подходящие методы моделирования.

1. Визуализация данных: Гистограммы: для распределения числовых переменных . Диаграммы разброса: для изучения взаимосвязи между двумя переменными
2. Разведочный анализ данны: Тип данных: числовые, категориальные, текстовые. Размерность: количество наблюдений и переменных. Пропущенные значения: количество и распределение.
3. Анализ закономерностей и аномалий: Определение тенденций: линейных, сезонных, циклических. Поиск выбросов: экстремальные значения, которые могут исказить результаты

[391]:
```
# Мы создаем Палитру
selec_palette = sns.color_palette("flare", n_colors = 10)
sns.palplot(selec_palette)
plt.show()
```

[392]:
```
sns.set_style('whitegrid')
sns.countplot(x='Компания',data=df_contrib, palette='flare')

plt.title("Типы и количества Компания")
```

[392]:
```
Text(0.5, 1.0, 'Типы и количества Компания')
```

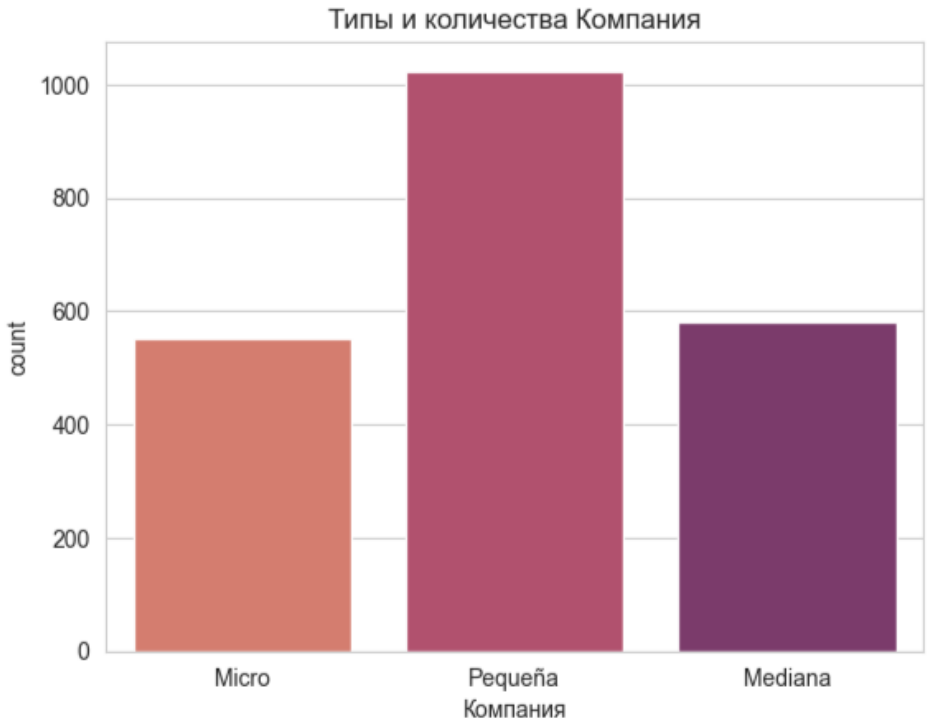


Рисунок 3 - график соотношения количеств типов MIPYMES

[393]:
```
# Мы создаем palette для проверки Нарушение
eva_color = ["#16a085", "#e74c3c"]

sns.palplot(eva_color)
plt.show()
```

[394]:
```
sns.set_style('whitegrid')
sns.countplot(x='Нарушение',data=auditoria, palette=eva_color,  saturation=0.80)
plt.title("Нарушение налогового")
```

[394]:
```
Text(0.5, 1.0, 'Нарушение налогового')
```

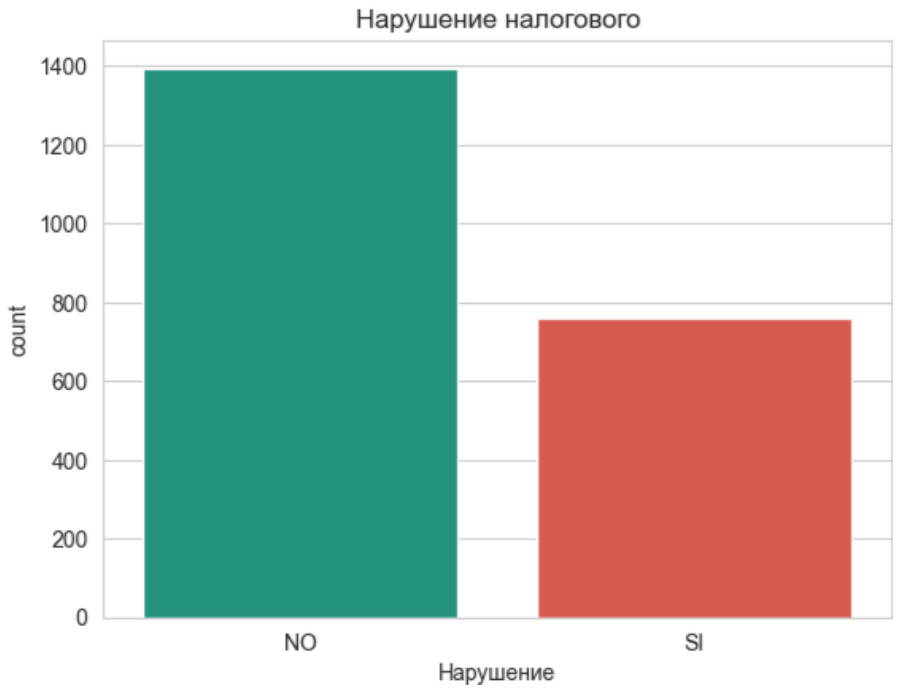


Рисунок 4 - график соотношения количества нарушений и отсутствия нарушений

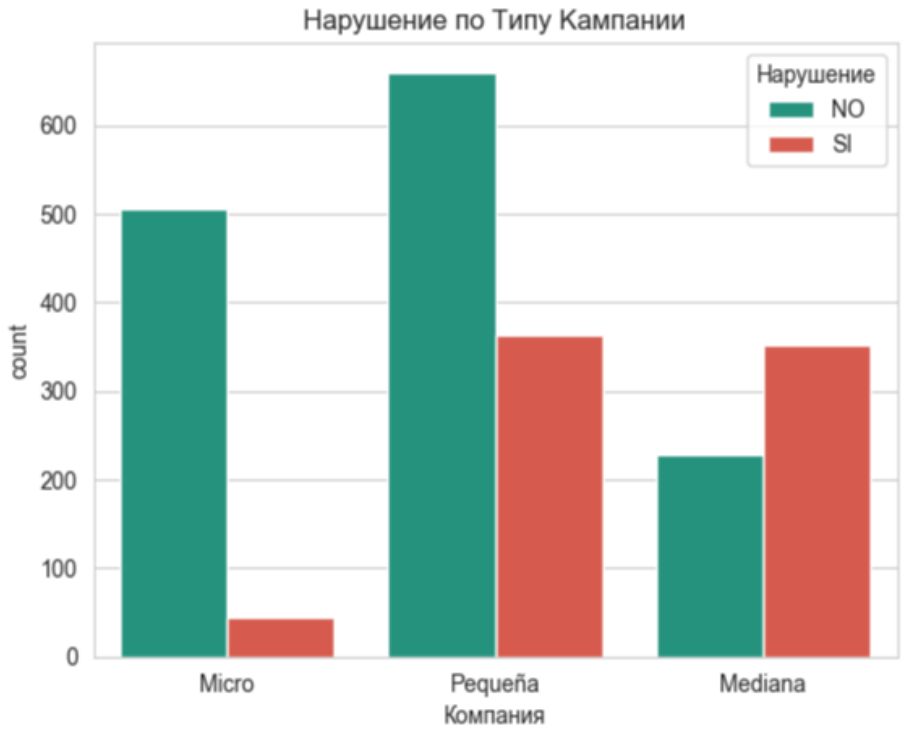
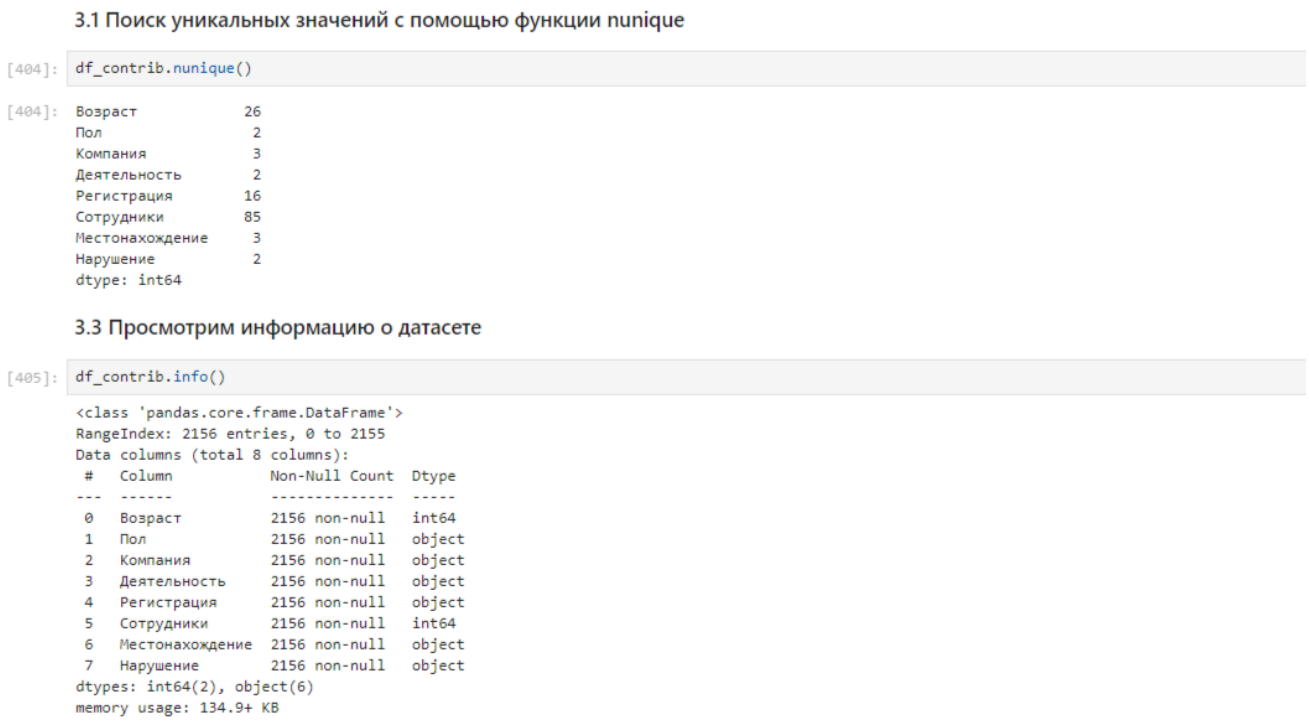


Рисунок 5 – график взаимосвязи между MIPYMES и налоговыми нарушениями

Рисунок 6 – график соотношения численности работников предприятия и налоговых нарушений

Перед передачей данных в модели машинного обучения их необходимо обработать и очистить. Очевидно, что грязные и необработанные данные могут содержать ошибки и пропущенные значения, что ненадежно, поскольку может привести к крайне неверным результатам моделирования. Но удалять что-то без какой-либо причины тоже неправильно. Вот почему должны сначала изучить набор данных..



3.1 Поиск уникальных значений с помощью функции nunique

```
[404]: df_contrib.nunique()

[404]: Возраст            26
       Пол                 2
       Компания            3
       Деятельность        2
       Регистрация        16
       Сотрудники         85
       Местонахождение     3
       Нарушение           2
       dtype: int64
```

3.3 Просмотрим информацию о датасете

```
[405]: df_contrib.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2156 entries, 0 to 2155
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Возраст          2156 non-null   int64
 1   Пол              2156 non-null   object
 2   Компания         2156 non-null   object
 3   Деятельность     2156 non-null   object
 4   Регистрация      2156 non-null   object
 5   Сотрудники       2156 non-null   int64
 6   Местонахождение  2156 non-null   object
 7   Нарушение        2156 non-null   object
dtypes: int64(2), object(6)
memory usage: 134.9+ KB
```

Рисунок 7 - описательная статистика датасета

Целью исследовательского анализа является получение первоначального представления о характере распределения переменных в исходном наборе данных, оценка качества исходных данных (пропуски, отклонения), определение характера взаимосвязи между переменными, а затем формулирование гипотез о наиболее важных из них.. подходящие модели машинного обучения для решения проблемы.

Данные объединённого датасета не имеют чётко выраженной зависимости, что подтверждает тепловая карта с матрицей корреляции и матрицы диаграмм рассеяния.

```
[417]: imp_ingreso.isnull().sum()
```

```
[417]: Январь       157
       Февраль      130
       Март          90
       Апрель        58
       Май           25
       Июнь           2
       Июль           0
       Август         0
       Сентябрь       0
       Октябрь        0
       Ноябрь         0
       Декабрь        0
       dtype: int64
```

```
[418]: #налог на продажу (imp_ingreso)
       sns.heatmap(imp_ingreso.isnull(),yticklabels=False,cmap=(selec_palette))
       #Тепловая карта, и функция ISNULL().. смотрим  что есть пропуска.
```

```
[418]: <Axes: >
```

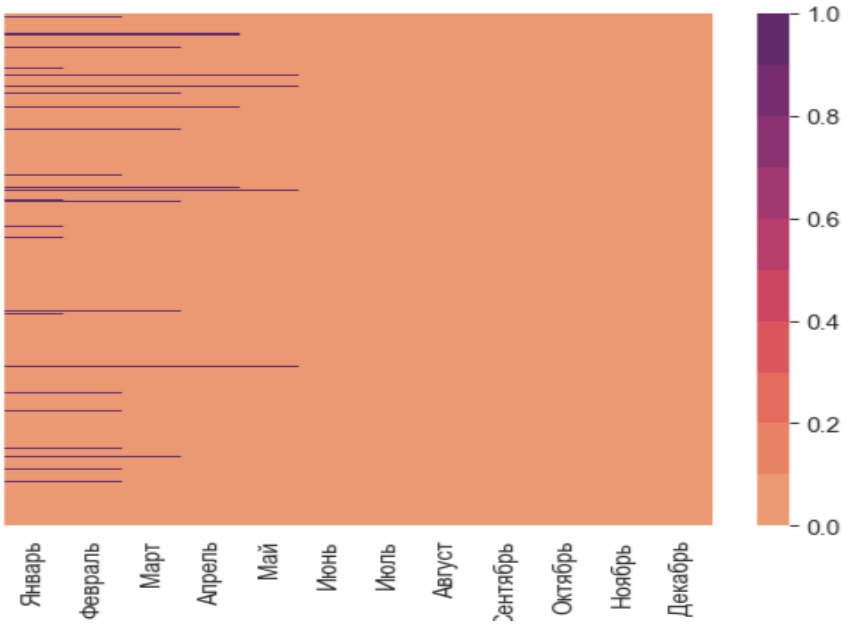


Рисунок 8 - Тепловая карта в поисках поисках в налогах на доходы компаний

```
rowss = imp_ingreso[imp_ingreso.isnull().any(axis=1)]

for index, row in rowss.iterrows():
    imp_ingreso = imp_ingreso.drop(index)
    imp_salario = imp_salario.drop(index)
    df_contrib = df_contrib.drop(index)
```

```
#налог на продажу (imp_ingreso)
sns.heatmap(imp_ingreso.isnull(),yticklabels=False,cmap=(selec_palette))
#Тепловая карта, и функция ISNULL().. смотрим  что есть пропуска.
```

```
<Axes: >
```

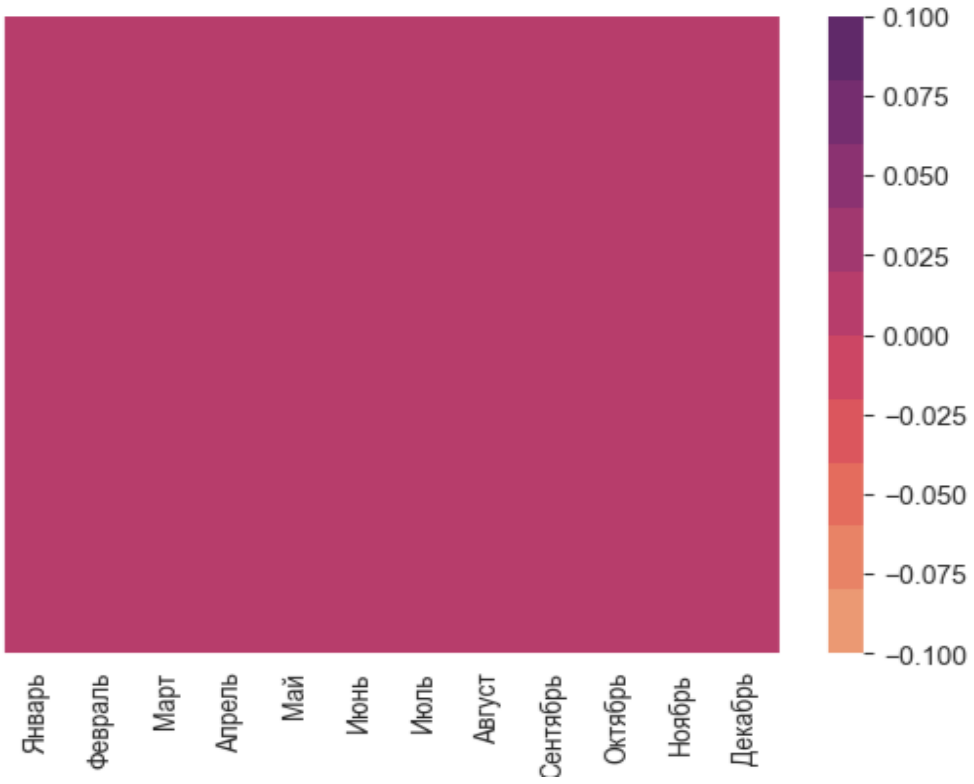


Рисунок 9 - Тепловая карта удаляем компании, которые не имеют всех налогов

```
suma_ingreso = imp_ingreso.sum(axis=1)
suma_salario = imp_salario.sum(axis=1)

suma_ingreso_df = pd.DataFrame(suma_ingreso,columns=['Налог_год'])

suma_salario_df = pd.DataFrame(suma_salario,columns=['Налог_за_Сотруд_год'])

df_contrib = pd.concat([df_contrib, suma_ingreso_df], axis=1)
df_contrib = pd.concat([df_contrib, suma_salario_df], axis=1)
```

```
df_contrib
```

	Возраст	Пол	Компания	Деятельность	Регистрация	Сотрудники	Местонахождение	Нарушение	Налог_год	Налог_за_Сотруд_год
0	41	1	0	1.0	9	5	0.0	0	2696187.0	9642.5
1	43	1	1	1.0	7	30	1.0	0	15848172.0	55440.0
2	47	1	1	0.0	8	31	1.0	1	14285684.0	22505.0
3	36	1	0	1.0	10	6	0.0	0	3363524.0	9660.0
4	42	1	1	0.0	13	22	0.0	0	12041140.0	41107.5
...	...	...	...	...	...	...	...	...	...	...
2150	37	1	1	0.0	14	11	1.0	0	6552778.0	20475.0
2152	44	1	1	0.0	15	34	0.0	0	17725927.0	71050.0
2153	42	1	1	1.0	11	23	0.0	1	10527362.0	21402.5
2154	31	1	1	1.0	11	14	0.0	0	8132127.0	22960.0
2155	35	1	1	0.0	7	21	1.0	1	9780844.0	24255.0

1999 rows × 10 columns

Рисунок 9 - Добавление дополнительных значений

Мы добавляем в набор данных исследования сумму значений уплата налогов за каждый месяц, чтобы уменьшить объем данных исследования таким образом, мы можем более четко оценить схемы оплаты для каждого налогоплательщики.

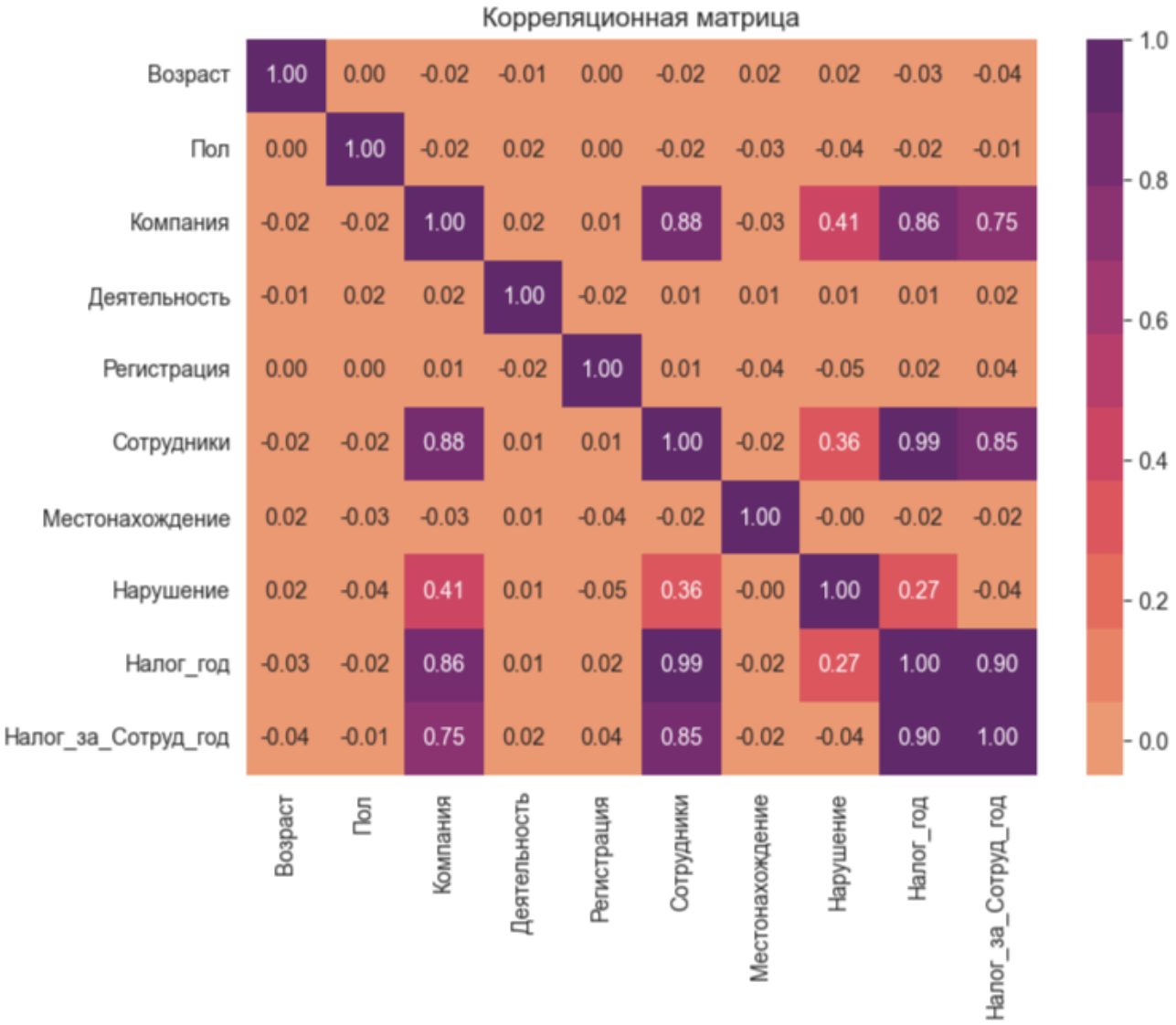


Рисунок 10 - тепловая карта с корреляцией данных

Отметим, что между некоторыми переменными существуют линейные зависимости, а другие корреляционные связи относятся к налоговому нарушению от 0,27 до 0,41

2. Практическая часть

2.1. Предобработка данных

Преобразуем столбец Нарушение в значения 0 или 1 «0» и «1».

```
# Приведем столбец Нарушение к значениям 0 и 1 и integer
df_contrib = df_contrib.replace({'Нарушение': {'NO' : 0, 'SI': 1}})
df_contrib['Нарушение'] = df_contrib['Нарушение'].astype(int)
```

Рисунок 11 - часть кода с преобразованием столбца " Нарушение"

По условиям задания применяем StandardScale. Это метод предобработки данных в машинном обучении, который масштабирует ваши данные, чтобы они имели нулевое среднее и единичную дисперсию.

применяем StandardScaler

```
565]: scaler = StandardScaler()
      scaler.fit(X)
      X = scaler.transform(X)
```

Рисунок 12 - часть кода с примение StandarScaler

2.2. Разработка и обучение модели

Разработка и обучение моделей машинного обучения проводились для выходного параметра “ Нарушение”: Для решения применяются все методы, описанные выше.

7.1 KNeighborsClassifier

```
Трекер_результатов = []
```

```
knn = KNeighborsClassifier(n_neighbors=3)
```

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo: {accuracy * 100}%" )

Трекер_результатов.append({accuracy,"KNeighborsClassifier "})
```

```
Precisión del modelo: 85.0%
```

7.2 DecisionTreeClassifier

```
model_DT = DecisionTreeClassifier()
```

```
model_DT.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred2 = model_DT.predict(X_test)
# точность на основе тестового набора данных
accuracy2 = accuracy_score(y_test, y_pred2)
print(f"Precisión: {accuracy2*100}%")

Трекер_результатов.append({ accuracy2,"DecisionTreeClassifier"})
```

```
Precisión: 96.5%
```

Рисунок 13 Запуск моделей машинного обучения

7.3 GaussianNB

```
683]: model_G = GaussianNB()
```

```
692]: model_G.fit(X_train, y_train)
```

```
692]: GaussianNB
      GaussianNB()
```

```
695]: y_pred3 = model_G.predict(X_test)

      accuracy3 = accuracy_score(y_test, y_pred3)
      print(f"Precisión: {accuracy3 * 100}%")

      Трекер_результатов.append({accuracy3 ,"GaussianNB "})
```

```
Precisión: 70.66666666666667%
```

7.4 RandomForestClassifier

```
696]: model_RF = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
697]: model_RF.fit(X_train, y_train)
```

```
697]: RandomForestClassifier
      RandomForestClassifier(random_state=42)
```

```
698]: y_pred4 = model_RF.predict(X_test)

      accuracy4 = accuracy_score(y_test, y_pred4)
      print(f"Precisión: {accuracy4 * 100}%")

      Трекер_результатов.append({accuracy4,"RandomForestClassifier"})
```

Рисунок 14 Запуск моделей машинного обучения

7.5 LogisticRegression

```
[699]: model_LG = LogisticRegression()

[700]: model_LG.fit(X_train, y_train)

[700]: LogisticRegression
       LogisticRegression()
```

```
[701]: y_pred5 = model_RF.predict(X_test)

       accuracy5 = accuracy_score(y_test, y_pred4)
       print(f"Precisión: {accuracy5 * 100}%")

       Трекер_результатов.append({accuracy5,"LogisticRegression"})

       Precisión: 97.5%
```

7.6 Support Vector Regression

```
[702]: modelSVR = SVR(kernel='rbf')

[704]: modelSVR.fit(X_train, y_train)
       y_pred = modelSVR.predict(X_test)
       print("Precision: {:.2f}".format(modelSVR.score(X_test, y_test)*100))
       accuracy6 =modelSVR.score(X_test, y_test)
       Трекер_результатов.append({accuracy6,"SVR"})

       Precision: 78.33
```

Рисунок 15 Запуск моделей машинного обучения

2.3. Тестирование модели

После обучения моделей точность этих моделей оценивалась на тренировочных и тестовых образцах. Сравнение используется в тестовых данных . Результат неудовлетворительный.

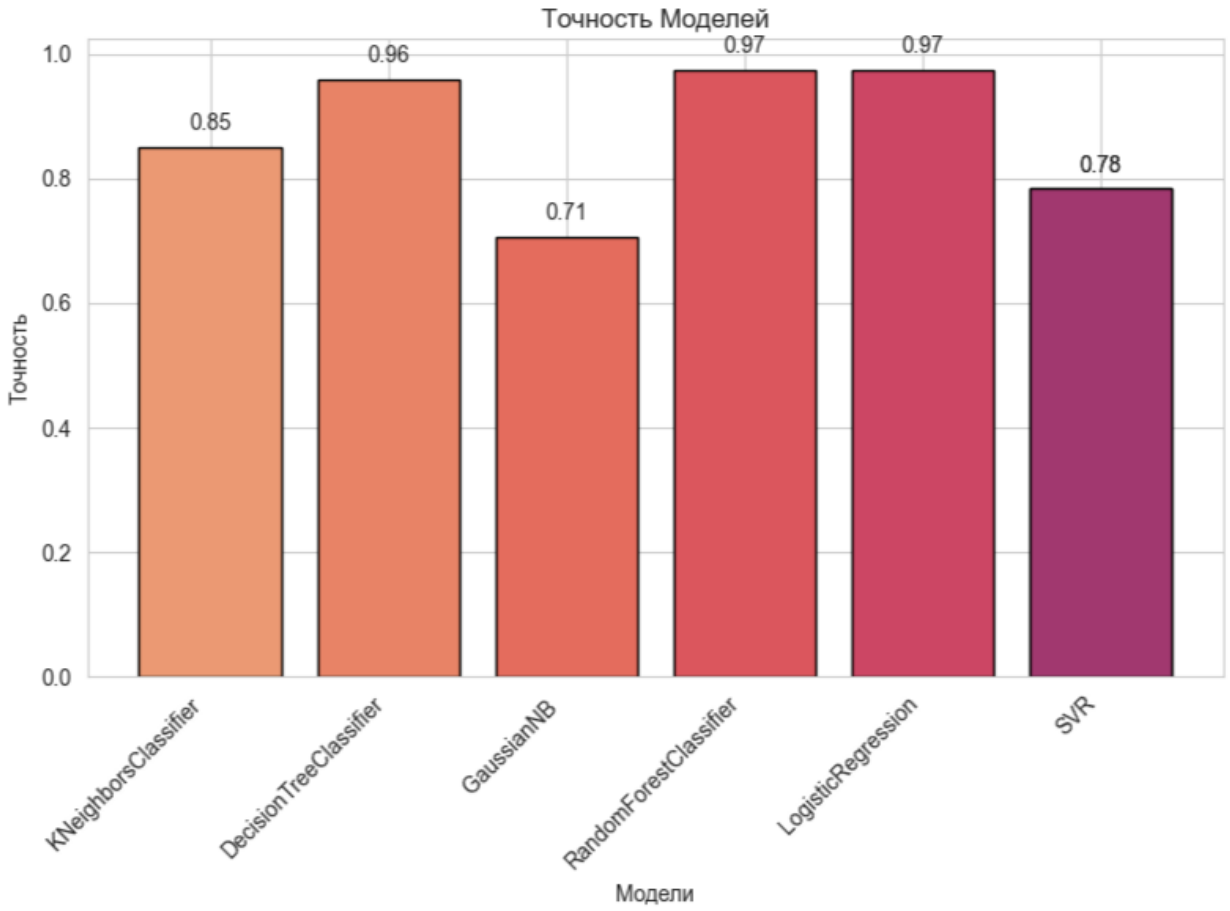


Рисунок 16 Точность моделей

2.4 Написать нейронную сеть

Обучение нейронной сети - это процесс, в ходе которого выбираются оптимальные параметры модели с точки зрения минимизации функциональности ошибок.Начнём стоить нейронную сеть с помощью класса keras.Sequential.

[706]:
```
X = X.astype('float32')
```

[707]:
```
y = LabelEncoder().fit_transform(y)
```

[708]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, shuffle = True)
```

[709]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1499, 9)
(500, 9)
(1499,)
(500,)
```

[710]:
```
n_features = X.shape[1]
```

[711]:
```
model = Sequential()
```

[165]:
```
model.add(Dense(20, activation = 'relu', input_shape = (n_features,) ))
model.add(Dense(10, activation = 'relu'))
#выходной слой
model.add(Dense(1, activation = 'sigmoid'))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy')
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#обучение
history = model.fit(X_train, y_train,
                    epochs = 20,
                    batch_size = 32,
                    verbose = 1,
                    validation_data = (X_test, y_test))

# predict test_set
yhat = (model.predict(X_test)>0.5).astype("int32")

#Валидируем прогноз
score = accuracy_score(y_test, yhat)
print('Accuracy: ', score)
```

Рисунок 17 первая нейронная сеть

Обучим и оценим модель.

```
83]: pyplot.title('Обучение')
     pyplot.xlabel('epochs')
     pyplot.ylabel('Cross_entropy')
     pyplot.plot(history.history['loss'], label = 'train')
     pyplot.plot(history.history['val_loss'], label = 'val')
     pyplot.legend()
     pyplot.show()
```

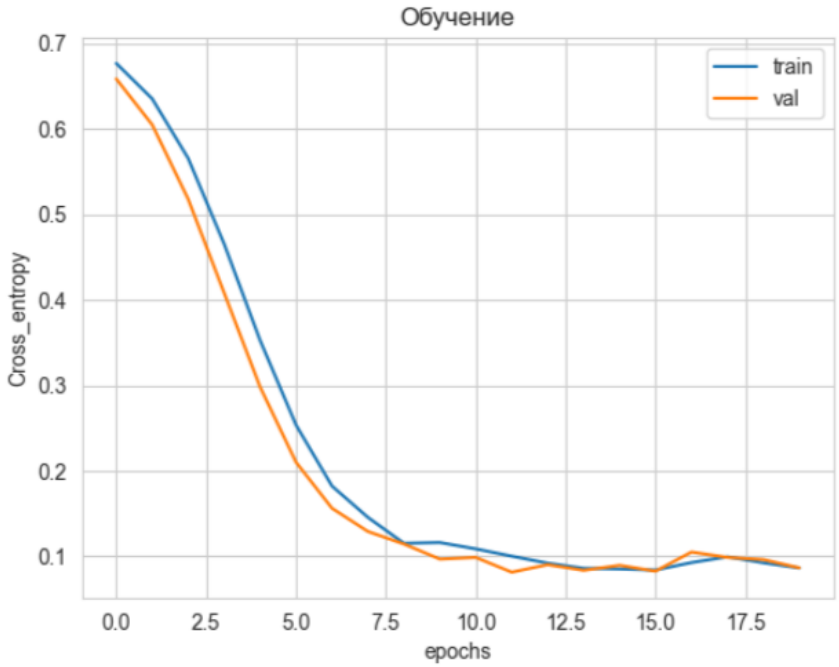


Рисунок 18 график потерь модели 1

[1684]:
```
model.add(Dense(32, activation = 'relu', input_shape = (n_features,) ))
model.add(Dense(32, activation = 'relu'))
#выходной слой
model.add(Dense(1, activation = 'sigmoid'))

#model.compile(optimizer = 'adam', loss = 'binary_crossentropy')
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#обучение
history = model.fit(X_train, y_train,
                    epochs = 20,
                    batch_size = 32,
                    verbose = 1,
                    validation_data = (X_test, y_test))

# predict test_set
yhat = (model.predict(X_test)>0.5).astype("int32")
#predictions = (model.predict(x_test) > 0.5).astype("int32")
#yhat = model.predict_classes(X_test)

#Валидируем прогноз
score = accuracy_score(y_test, yhat)
print('Accuracy: ', score)
```

```
Epoch 1/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8608 - loss: 0.6682 - val_accuracy: 0.9800 - val_loss: 0.5825
Epoch 2/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9844 - loss: 0.5474 - val_accuracy: 0.9820 - val_loss: 0.4439
Epoch 3/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9799 - loss: 0.4026 - val_accuracy: 0.9820 - val_loss: 0.2969
Epoch 4/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9771 - loss: 0.2616 - val_accuracy: 0.9820 - val_loss: 0.1821
Epoch 5/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9782 - loss: 0.1665 - val_accuracy: 0.9820 - val_loss: 0.1188
Epoch 6/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9762 - loss: 0.1248 - val_accuracy: 0.9820 - val_loss: 0.0943
Epoch 7/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9831 - loss: 0.0896 - val_accuracy: 0.9820 - val_loss: 0.0859
Epoch 8/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9842 - loss: 0.0804 - val_accuracy: 0.9820 - val_loss: 0.0836
Epoch 9/20
```

Рисунок 19 - создание второй модели

```
Epoch 18/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9841 - loss: 0.0748 - val_accuracy: 0.9820 - val_loss: 0.0830
Epoch 19/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9827 - loss: 0.0802 - val_accuracy: 0.9820 - val_loss: 0.0831
Epoch 20/20
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9859 - loss: 0.0685 - val_accuracy: 0.9820 - val_loss: 0.0818
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
Accuracy:  0.982
```

[1685]:
```
pyplot.title('Обучение')
pyplot.xlabel('epochs')
pyplot.ylabel('Cross_entropy')
pyplot.plot(history.history['loss'], label = 'train')
pyplot.plot(history.history['val_loss'], label = 'val')
pyplot.legend()
pyplot.show()
```

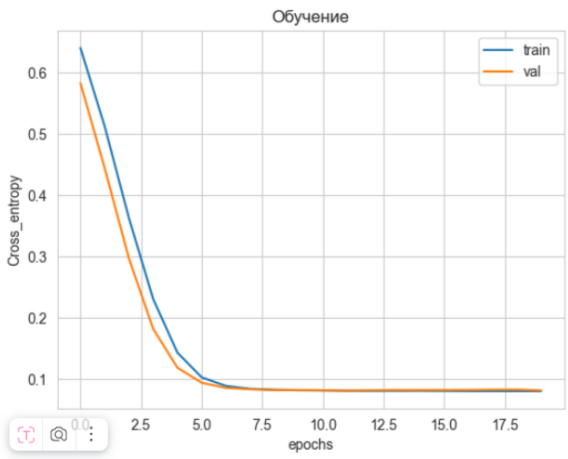


Рисунок 20 - график потерь второй модели

2.5 Создание удалённого репозитория и загрузка

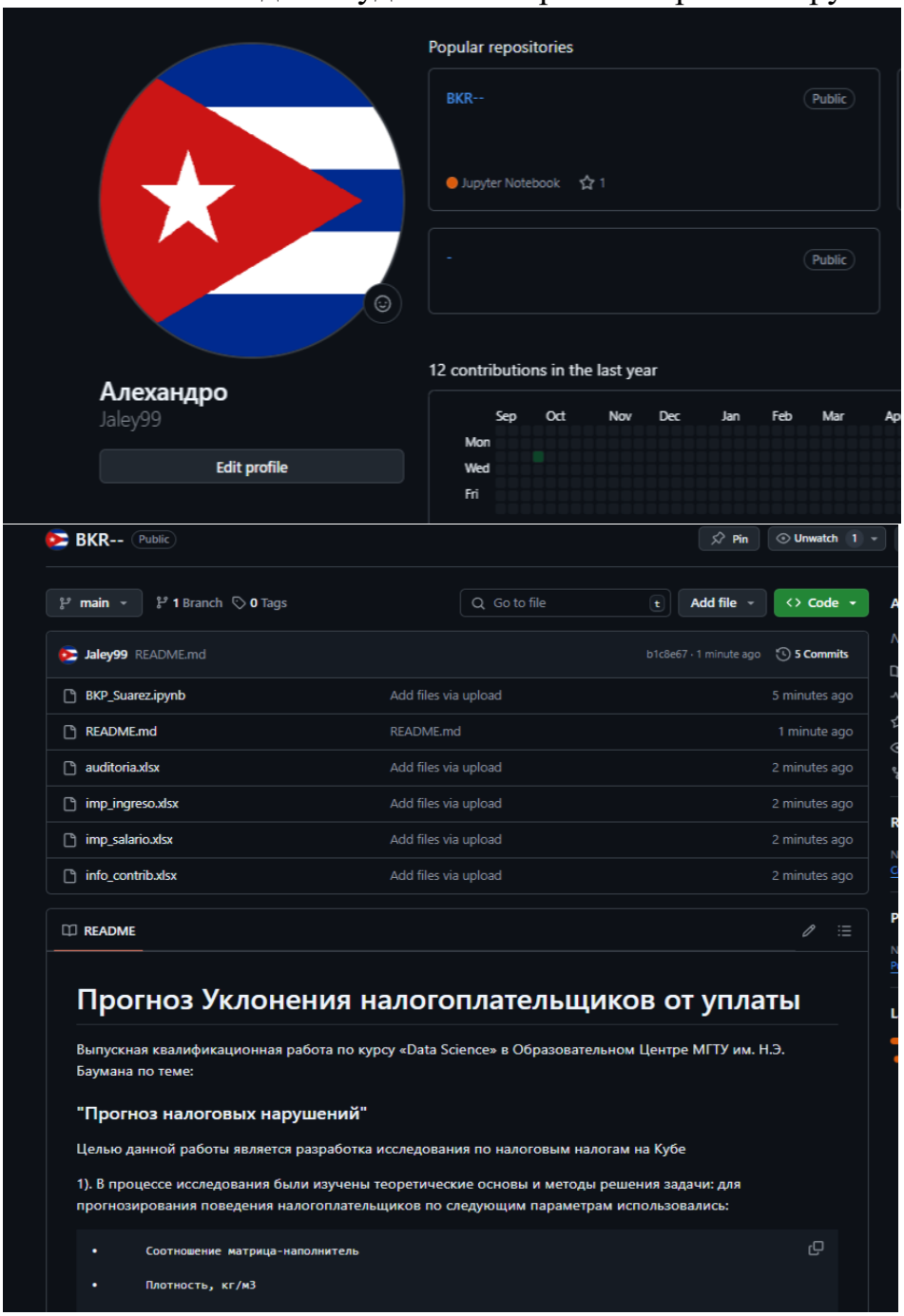


Рисунок 21 - часть страницы на github.com файла и файла README

2.6 **Заключение**

Эта исследовательская работа позволяет сделать некоторые основные выводы по этому вопросу. Распределение полученных данных в объединенном наборе данных близко к нормальному, коэффициенты корреляции между переменными демонстрируют линейную зависимость между некоторыми переменными и "НАРУШЕНИЕМ". Подходы, использованные при разработке моделей, привели к надежным прогнозам. Примененные регрессионные модели показали высокую эффективность прогнозирования. Наилучшие показатели точности были примерно на уровне 98 процентов точности. Был сделан вывод о возможности определения с помощью высокого порогового значения возможности уклонения налогоплательщиков от уплаты налогов. Для более глубокого изучения проблемы уклонения от уплаты налогов нам понадобятся данные, в которых мы сможем проанализировать зависимость деятельности компаний и налогоплательщиков на Кубе от уклонения от уплаты налогов.

2.7 Список используемой литературы

1. Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – Режим доступа: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19. (дата обращения: 07.06.2022)

2. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.

3. Абу-Хасан Махмуд, Масленникова Л. Л.: Прогнозирование свойств композиционных материалов с учётом наноразмера частиц и акцепторных свойств катионов твёрдых фаз, статья 2006 год

4. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.

5. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Издательство Казанского университета, 2018. – 121 с.

6. Документация по библиотеке keras: – Режим доступа: https://keras.io/api/.(дата обращения: 08.06.2022).

7. Документация по библиотеке matplotlib: – Режим доступа: https://matplotlib.org/stable/users/index.html. (дата обращения: 10.06.2022)

8. Документация по библиотеке numpy: – Режим доступа: https://numpy.org/doc/1.22/user/index.html#user. (дата обращения: 03.06.2022).

9. Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html#user-guide. (дата обращения: 04.06.2022).

10. Документация по библиотеке scikit-learn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html. (дата обращения: 05.06.2022).

11. Документация по библиотеке seaborn: – Режим доступа: https://seaborn.pydata.org/tutorial.html. (дата обращения: 06.06.2022).

12. Документация по библиотеке Tensorflow: – Режим доступа: https://www.tensorflow.org/overview (дата обращения: 10.06.2022).

13. Документация по языку программирования python: – Режим доступа: https://docs.python.org/3.8/index.html. (дата обращения: 02.06.2022).

14. Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: https://habr.com/ru/post/428503/ (дата обращения 07.06.2022)

15. Миронов А.А. Машинное обучение часть I ст.9 – Режим доступа: http://is.ifmo.ru/verification/machine-learning-mironov.pdf. (дата обращения 08.06.2022)

16. Плас Дж. Вандер, Python для сложных задач: наука о данных и машинное обучение. Санкт-Петербург: Питер, 2018, 576 с.

17. Реутов Ю.А.: Прогнозирование свойств полимерных композиционных материалов и оценка надёжности изделий из них, Диссертация на соискание учёной степени кандидата физико-математических наук, Томск 2016.

18. Руководство по быстрому старту в flask: – Режим доступа: https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html. (дата обращения: 09.06.2022)

19. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.

20. Скиена, Стивен С. С42 Наука о данных: учебный курс.: Пер. с англ. - СПб.: ООО "Диалектика", 2020. - 544 с. : ил.

2.8 Приложение 1

Подробный план работы:

1. Загружаем и обрабатываем входящие датасеты
   1.1 Переименовываем столбцы
   1.2 Удаляем неинформативные столбцы
   1.3 Объединяем датасеты по методу INNER

2 Визуализация полей датасет
   1.4 Изучим описательную статистику и мы создаем графики
   1.5 Построим несколько вариантов гистограмм распределения каждой переменной

3 Проводим разведочный анализ данных

   3.5 Проверим датасет на пропуски
   3.6 Мы преобразуем категориальные переменные ('Пол', 'Компания', 'Деятельность', 'Регистрация', 'Местонахождение', 'Нарушение') в числовой формат, подходящий для моделей машинного обучения.
   3.7 Мы применяем LabelEncodeer и OrdinalEndocer к категориальным переменным

4 Проверка значений
   4.1 Тепловая карта смотрим что есть пропуска
   4.2 Удаляем компании, которые не имеют всех налогов

5 Добавление дополнительных значений
   5.1 Построим несколько вариантов попарных графиков рассеяния точек (матрицы диаграмм рассеяния)
   5.2 Построим Корреляционная матрица
   5.3 Построим корреляционную матрицу с помощью тепловой карты

6 Проведём предобработку данных (в данном пункте только очистка датасета от выбросов)

6.1 Определяем объектные переменны

6.2 Проведём стандартизацию (продолжим предобработку данных)

7 Разработаем и обучим нескольких моделей прогноза прочности при растяжении (с 30% тестовой выборки)

7.1 Определим входы и выходы для моделей

7.2 Разобьём данные на обучающую и тестовую выборки

8 Построим и визуализируем результат работы метода опорных векторов

8.1 KneighborsClassifier

8.2 DecisionTreeClassifier

8.3 GaussianNB

8.4 RandomForestClassifier

8.5 LogisticRegression

8.6 SVR

8.7 Точность моделей Машинного Обучения

9 Нейронная сеть

9.1 Нормализуем данные

9.2 Построим модель, определим параметры

9.3 Посмотрим на результаты

9.4 Обучим нейросеть 80/25

9.5 Оценим модель

9.6 Посмотрим на потери модели

9.7 Посмотрим на график потерь на тренировочной и тестовой выборках

9.8 Сконфигурируем другую модель, зададим слои

9.9 Посмотрим на потери другой модели

9.10 Посмотрим на график потерь на тренировочной и тестовой выборках

10 Создание удалённого репозитория и загрузка результатов работы на него.

10.1 https://github.com/Jaley99/BKR--

10.2 Создадим README (https://github.com/Jaley99/BKR--/blob/main/README.md)

10.3 Выгрузим все необходимые файлы в репозиторий