

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc186163388)

[1.**АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 4](#_Toc186163389)

[1.1. Знакомство с платформой для разработки на языке Python 4](#_Toc186163390)

[1.2. Общая информация про нейронные сети 5](#_Toc186163391)

[1.3. Виды нейронных сетей 5](#_Toc186163392)

[1.4. Математическая модель нейронной сети 6](#_Toc186163393)

[1.5. Заключение по разделу 7](#_Toc186163394)

[2.**СОЗДАНИЕ НЕЙРОННОЙ СЕТИ** 8](#_Toc186163395)

[2.1. Создание нейронной сети и нейронных блоков 8](#_Toc186163396)

[2.2. Сбор и нормализация данных для обучения нейронной сети 9](#_Toc186163397)

[2.3. Расчет количества потерь и их минимизации 12](#_Toc186163398)

[2.4. Анализ адекватности обучения 12](#_Toc186163399)

[2.5. Оценка важности признаков после обучения 16](#_Toc186163400)

[2.6. Заключение по разделу 18](#_Toc186163401)

[**ЗАКЛЮЧЕНИЕ** 19](#_Toc186163402)

[**СПИСОК ЛИТЕРАТУРЫ** 20](#_Toc186163403)

# **ВВЕДЕНИЕ**

Производственная практика проходила в Московском университете им. С.Ю. Витте на кафедре информационных технологий, где во время учебного процесса занимаюсь решением поставленной задачи, а именно применение нейронных сетей для кредитного скоринга.

Кредитный скоринг – это оценка которая помогает банкам и другим финансовым учреждениям принимать решения для выдачи кредитов и управлении рисками. Современные технологии такие как машинное обучение и нейронные сети значительно улучшают точность и эффективность кредитный решений.

Применяя нейронные сети в кредитном скоринге процесс оценки кредитоспособности автоматизируется, сокращается время принятия решения и улучшается управление кредитными рисками. Полезность нейронных сетей проявляется в том, что они учитывают широкий спектр факторов, такие как демографические данные, истории кредитных операций и многое другое, это дает возможность создавать точные и надежные модели.

Целью практической работы является применение и изучение нейронных сетей для задачи кредитного скоринга, разработка и реализация модели нейронной сети, оценка производительности модели, оценка вклада разных признаков и анализ влияния на выдачу кредита.

Что бы достичь целей практики, нужно выполнить следующие задачи.

Нужно выбрать платформу для реализации задачи, проанализировать набор данных, рассмотреть связи между ними, провести предварительную подготовку данных, разделить данные на обучающие и тестовые, реализовать и обучить модель, провести ее оценку с визуализацией результатов.

# 1.**АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

1.1. Знакомство с платформой для разработки на языке Python

Google Colab

Мой выбор Google Colab, потому что это бесплатная платформа для работы на языке Python прямо в браузере для повышения навыков и исследований в машинном обучении. Его не нужно настраивать, нужно лишь подключение к интернету, все работы сохраняются в облаке Google, можно так же работать совместно с людьми и делиться результатами.

Код можно прописывать в отдельных ячейках, это дает возможность выполнять код по порядку и просматривать каждый блок на наличие ошибок или результатов, а его интерфейс очень схож с Jupyter Notebook. Colab очень прост в использовании, у него понятный интерфейс, что делает его доступным для новый пользователей. В нем можно использовать огромное количество разных библиотек языка Python, такие как NumPy, pandas, Matpotlib, sklearn и другие.

Для своей работы я решил выбрать именно эту платформу, так как в ней очень удобно строить графики, диаграммы, изучать и работать с набором данных.

Google Colab — это мощный и удобный инструмент для работы с Python в облаке. Он подходит для тех, кто хочет экспериментировать, изучать и разрабатывать с анализом данных и машинным обучением в целом, особенно с бесплатным доступом ко всем вычислительным ресурсам.

## 1.2. Общая информация про нейронные сети

Нейронная сеть – это модель, которая при помощи компьютерного алгоритма способна обрабатывать большие объёмы данных, основа нейронных сетей, это обучение, она способна обучаться и распознавать сложные закономерности в данных, имитируя человеческий мозг. Нейронные сети могут играть в игры и решать разного рода головоломки, обрабатывать языки, распознавать образ и картинки и многое другое.

Основные компоненты нейронных сетей это – нейроны, они представляют из себя вычислительные единицы, они получают входные данные и производят их обработку, а затем передают результаты выше в синапсы(связи). Синапсы в свою очередь соединяются нейронами, у этих нейронов есть вес, который и определяют силу связи. Так же есть слои, они представляют из себя группу нейронов, объединенные по уровню обработки. Функция активации при помощи нелинейной функции в зависимости от полученного сигнала определяет будет ли нейрон активирован.

## 1.3. Виды нейронных сетей

Существует несколько видов нейронных сетей, которые предназначены для решения разного типа задач.

1. Многослойный перцептрон, полносвязная сеть, которая имеет несколько скрытых слоев связанных между собой. У нее простая архитектура, но требуется настройка с количеством скрытых слоев. Особенно хорошо работает с табличными данными, но может тяжело справляться с пространственными данными. Может быть подвержена переобучению, при слишком большом количестве нейтронов или большой глубине. Так же требуется предварительная обработка данных.
2. Рекуррентные нейронная сеть, предназначается для последовательных данных, например, текста или временных рядов. У такого типа сети есть память и она учитывает входы на основе предыдущего. Есть проблемы с затухающими или взрывающимися градиентами, особенно при очень длительных последовательностях. Требует хорошего структурирования данных.
3. Свёрточные нейронные сети, разработана специально для данных имеющих пространственную структуру, например, видео или картинки. Могут выявлять закономерности в данных, затрудняют работу, за счет сложности настройки и требовательности к вычислительным ресурсам. Требуют обработки данных.
4. Генеративно – состязательная сеть, сеть которая состоит из двух частей: генератора и дискриминатора, у генератора задача создавать новые данные, которые похожи на обучающие, а дискриминатора отличать сгенерированные данные от реальных. Обучение может быть сложным и нестабильным, требовательна к настройке всех параметров, так же из-за этого возможна нестабильность при обучении. Функции потерь имеет большое значение, поэтому нужно пробовать разные варианты. Чтобы стабилизировать сеть нужно использовать разные методы обучения. Оценить качество полученной модели может быть сложно.

1.4. Математическая модель нейронной сети

**Модель нейрона (перцептрона)**

Входные данные: X = [X1;X2;…Xn] – это вектор входных данных, а n – это количество признаков .

Веса это W : [W1….Wn] – вектор весов соответствует каждому входному признаку.

b – скалярное смещение (bias).   
Линейное преобразование: z = W1X2 + W2X2 +…. WnXn + b или в векторной форме: z = W \* X + b, где \* скалярное произведение.

Функция активации выглядит как: а = f(z) – применяемая к полученному линейному образованию.

**Многослойный перцептрон(MLP)**

I – номер слоя, где I = 0, это выходной слой

a^(I) – выходные значения

W^(I) – матрица весов

B^(I) – вектор смещения слоя

f^(I) – функция активации слоя

Линейное преобразование: z^(I) = W^(I) \* a^(I-I) + b^(I), где a^(I-I) это выходные значения предыдущего слоя, матрица весов это W^(I), вектор смещений b^(I).

**Рекуррентные нейронные сети**

Обновление скрытого состояния: ht = f(W\_xh \* xt + W\_hh \* ht-1 + b\_h)

W\_xh – матрица весов входа

W\_hh – матрица весов для скрытого состояния

b\_h – вектор смещения для скрытого состояния

f – функция активации

ht-1 – скрытое состояние на предыдущем временном шаге

xt – вход на текущем временном шаге

Вычисление выхода: yt = g(W\_yh \*ht + b\_y)

W\_yh – матрица весов для входа

B\_y – вектор смещения для выхода

g – функция активации(пример сигмоида для бинарной классификации)

## 1.5. Заключение по разделу

В рамках практической работы мы выбрали Google Colab, так как это удобная платформа для обучения моделей, познакомились с видами нейронных сетей и их основами, так же рассмотрели математические модели, это позволило получше углубиться в устройство этих алгоритмов. Полученные теоретические знания будут использованы для выполнения практической работы, а именно, создание и оценке модели нейронной сети для кредитного скоринга.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Выводы | Сформированные компетенции | |
| Код компетенции | Содержание компетенции |
| Проведен выбор и развертывание среды разработки для языка программирования Python. | **ПК-1** | Обладать способностью разработки прикладного программного обеспечения, автоматизации работы с базами данных и документами, программирования бизнес-логики приложений, интеграции разнородных данных |
| Проведен анализ теоретических аспектов создания нейросетей  Разработана математическая модель нейронной сети | **ПК-8** | Знать методы и инструментальные средства интеллектуального анализа больших данных |

# 2.**СОЗДАНИЕ НЕЙРОННОЙ СЕТИ**

## 2.1. Создание нейронной сети и нейронных блоков

Создаем последовательную модель Sequential, это говорит о том, что она добавляет слои последовательно, все данные будут проходить через каждый слой по порядку.

model = Sequential()

model.add(Dense(100, activation='relu', input\_shape=(X\_train.shape[1],)))

model.add(Dropout(0.2))

model.add(Dense(50, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

Создаем первый скрытый слой для нейронной сети. Dense – это добавление полносвязного слоя со 100 нейронами, где каждый нейрон связан с входными признаками. Далее идет функция активации relu и применяется к каждому выходу нейронов, а X\_train.shape получает количество столбцов в обучающем наборе, для того что бы нейронная сеть знала, какой формы будут входные данные. Далее создаем слой Dropout что бы во время обучения 20% нейронов отключались случайным образом, это делает для того чтобы предотвратить обучение, делая модель более устойчивой к разного рода изменениям во время обучения. Потом мы добавляем второй слой с меньшим количеством нейронов, где все эти нейроны связаны со всеми нейронами предыдущего слоя, это позволит модели выявить более сложные закономерности. Добавляем еще один Dropout для лучшего регулирования. В конце мы создаем слой с одним нейроном, он и будет выдавать конечное предсказание, а функция активации сигмоиды преобразует результат значения от 0 до 1.

## 2.2. Сбор и нормализация данных для обучения нейронной сети

Сбор данных происходит на этапе считывания датасета. Рисунок 1.

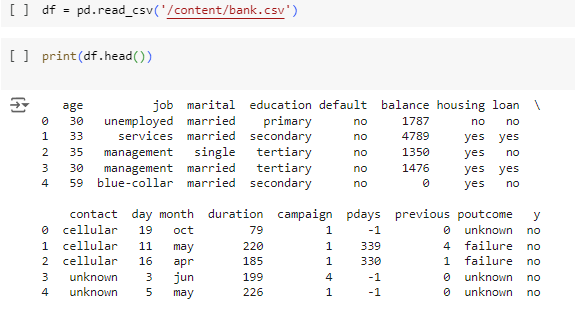


Рисунок 1. Данные датасета

numerical\_features = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']

categorical\_features = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'poutcome', 'month']

На данном фрагменте кода показано разделение на числовые и категориальные признаки, это важный шаг для подготовки данных. Так же устанавливаются связи между ними. Пример рисунков 2, 3 и 4.

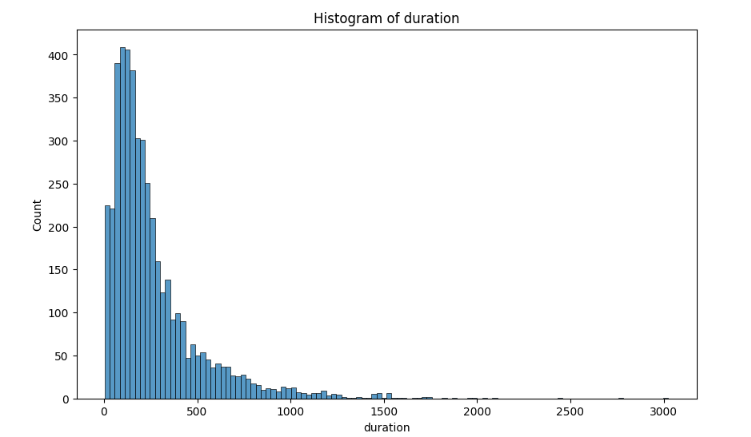
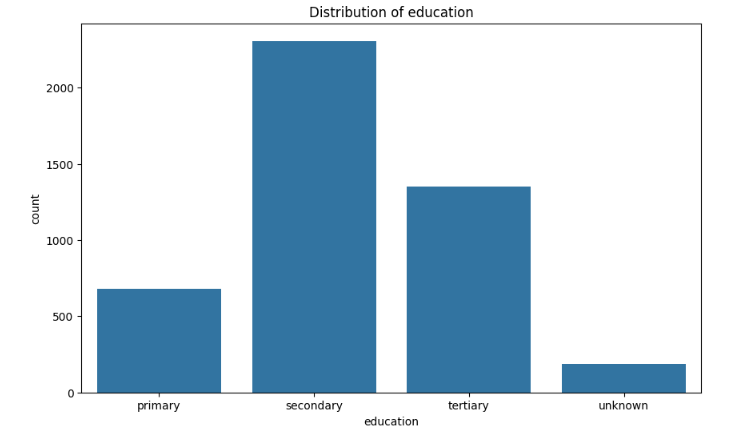


Рисунок 2. Числовые данные

Рисунок 3. Категориальные данные

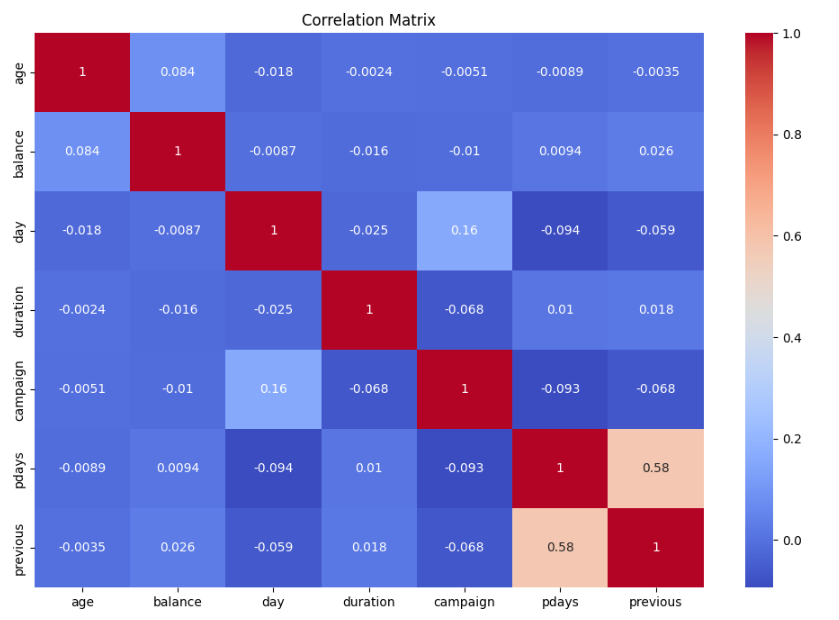


Рисунок 4. Корреляционная матрица

Корреляционная матрица хорошо показывает какие переменные могут быть важны для анализа в будущем, так как она точно указывает где есть корреляция и где ее нет.

На данном этапе мы разделяем признаки и целевую переменную.  
Признаки это входные данные нейронной сети, а целевая переменная это то, что мы должны предсказать.

X = df.drop(columns=['y'])

y = df['y']

Следующим этапом маштабируем числовые признаки. Это делается для того, чтобы помочь нейронам сети обучаться быстрее и стабильнее. Это один из важнейших этапов нормализации данных.

numerical\_cols = X.select\_dtypes(include=np.number).columns

scaler = StandardScaler()

X[numerical\_cols] = scaler.fit\_transform(X[numerical\_cols])

Разделяем данные на тестовую и обучающую выборки, это делается для обучения модели и для оценки ее дальнейшей производительности.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

## 2.3. Расчет количества потерь и их минимизации

Расчет происходит путем использования оптимизатора adam и binary\_crossentropy, выбор этих команд минимизирует потери.

model.compile(optimizer='adam',

              loss='binary\_crossentropy',

              metrics=['accuracy'])

Обучение нейронной сети

early\_stopping = EarlyStopping(monitor='val\_loss', patience=10)

history = model.fit(X\_train, y\_train,

                    epochs=200,

                    validation\_data=(X\_test, y\_test),

                    callbacks=[early\_stopping],

                    verbose=1)

Создаем объект для управления процессом обучения(EarlyStopping), он выполняет механизм ранней остановки, делается это для просмотра функций потерь на проверочных данных, то есть если метрика не улучаешься в течении 10 эпох из 200, то обучение останавливается. В данном случае val\_loss метрика для отслеживания, означает она потери на проверочной выборке. Если все успешно, то модель обучается, используя обучающие данные и проверяет результаты на проверочных данных по каждой эпохе.

## 2.4. Анализ адекватности обучения

y\_pred\_prob = model.predict(X\_test).flatten()

y\_pred = np.where(y\_pred\_prob > 0.5, 1, 0)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nROC AUC score:\n", roc\_auc\_score(y\_test, y\_pred\_prob))

r2 = r2\_score(y\_test, y\_pred)

print(f"R2 Score: {r2:.4f}")

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(7, 7))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

Данный код оценивает обученную модель. Прогнозирует вероятности тестовых данных и преобразует их в бинарные классы, далее выводит отчет о классификации и нужных показателей, а также матрицу ошибок. Рисунки 5 и 6.

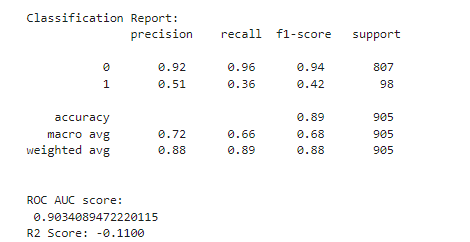


Рисунок 5. Отчет классификации

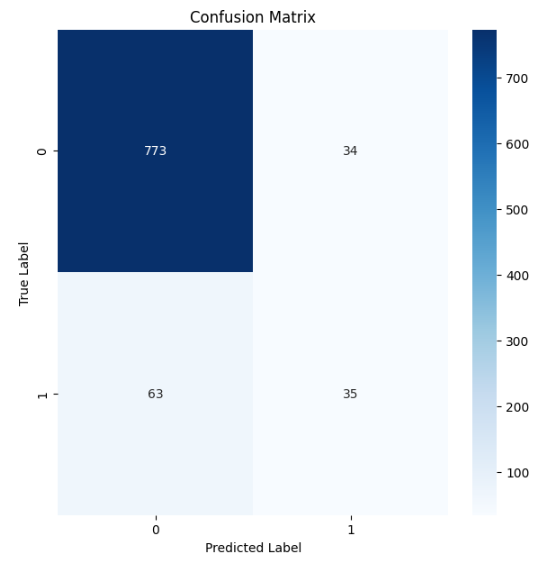


Рисунок 6. Матрица ошибок

Из увиденного можно сделать вывод, что 0 класс модель предсказывает хорошо, но класс 1 распознает плохо, об этом нам говорит низкие показатели точности и полноты. У модели общая точность хорошая 89% и она хорошо разделяет классы, об этом нам говорит ROC AUC score.

Дополнительная визуализация модели. Рисунки 7, 8 и 9.

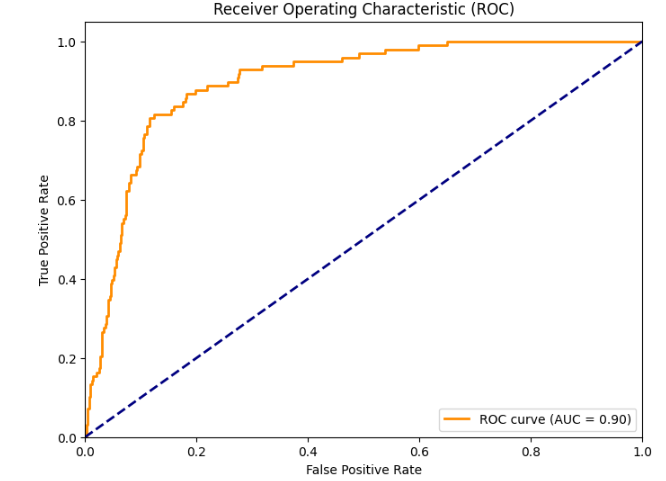


Рисунок 7. ROC – кривая

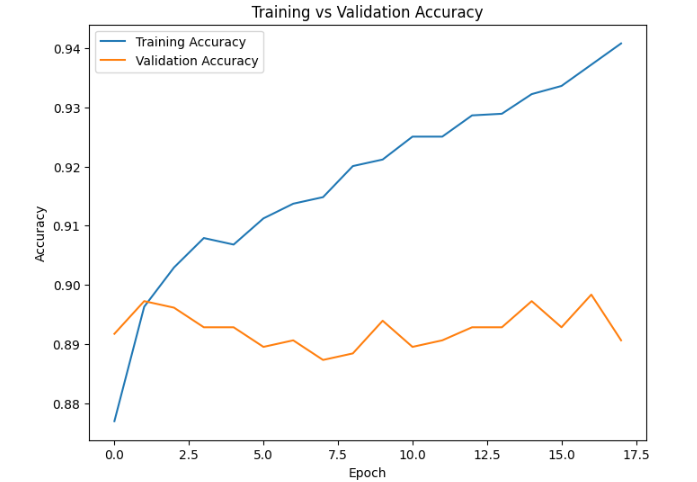


Рисунок 8. Сравнение точности обучения и проверки.

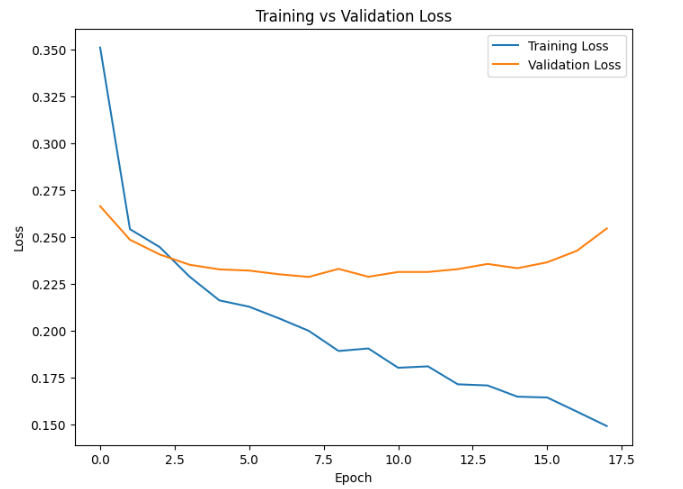


Рисунок 9. Сравнение обучения и валидации потерь

## 2.5. Оценка важности признаков после обучения

class KerasClassifierWrapper(BaseEstimator, ClassifierMixin):

    def \_\_init\_\_(self, model):

        self.model = model

    def fit(self, X, y):

        return self

    def predict(self, X):

         y\_pred\_prob = self.model.predict(X).flatten()

         return np.where(y\_pred\_prob > 0.5, 1, 0)

    def score(self, X, y):

        y\_pred\_prob = self.model.predict(X).flatten()

        y\_pred = np.where(y\_pred\_prob > 0.5, 1, 0)

        return accuracy\_score(y, y\_pred)

Создаем класс – оболочку для модели Keras, совмещая ее с scikit, чтобы в будущем сделать анализ важности признаков модели.

model\_wrapper = KerasClassifierWrapper(model)

result = permutation\_importance(model\_wrapper, X\_test, y\_test, n\_repeats=10, random\_state=42)

importance = pd.Series(result.importances\_mean, index=X\_test.columns)

importance\_sorted = importance.sort\_values(ascending=False)

Модель оборачивается в класс KerasClassifierWrapper, для вычисления важности признаков на тестовых данных. Происходит это путем определения важности каждого признака путем случайных перестановок значений признака, а также наблюдением за тем, как это влияет на производительность модели. Чем больше падает точность, тем важнее признак.

top\_n = 5

print("\nTop",top\_n,"Наиболее важных особенностей:\n", importance\_sorted.head(top\_n))

Выводим 5 наиболее важных признаков после обучения модели. Рисунок 10.

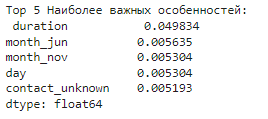


Рисунок 10. Важность признаков

## 2.6. Заключение по разделу

В результате проделанной работы по обучению модели было показано, что нейронная сеть способна решать задачу классификации, но требует доработки по устранению дисбаланса классов, а также более детальный анализ модели. Общую картину процесса обучения и его результатов дала визуализация данных.  
Таблица 2.

|  |  |  |
| --- | --- | --- |
| Выводы | Сформированные компетенции | |
| Код компетенции | Содержание компетенции |
| Реализованы алгоритмы сбора информации и создания информационных объектов на Python | **ПК-1** | Обладать способностью разработки прикладного программного обеспечения, автоматизации работы с базами данных и документами, программирования бизнес-логики приложений, интеграции разнородных данных |
| Разработаны нейроны и нейронные блоки на языке программирования Python.  Создана нейронная сеть. Экспериментальным способом подобраны характеристики нейронной сети: определено количество слоев, число блоков в скрытых слоях, наличие или отсутствие обходных соединений, передаточные функции нейронов.  Собраны данные для обучения нейронной сети. Параметры обучения подобраны экспериментальным способом.  Произведено обучение нейронной сети на собранных данных. Проверена адекватность результатов обучения сети. | **ПК-8** | Знать методы и инструментальные средства интеллектуального анализа больших данных |

# **ЗАКЛЮЧЕНИЕ**

В ходе данной практической работы была выбрана платформа на которой была построена и обучена нейронная сеть, для задачи кредитного скоринга. Во время выполнения работы была произведена обработка данных, был проведен анализ модели и сделаны выводы по каждому разделу. Анализ важности признаков показал какие факторы наиболее значимые, они могут быть использованы для повышения качества прогнозов и оптимизации бизнес - процессов. Были получены новые навыки в сфере машинного обучения и опыт работы с нейронными сетями, который я смогу использовать в своих дальнейших работах.

# **СПИСОК ЛИТЕРАТУРЫ**

1. “Основы Data Science и машинного обучения” автор Гусев Е. (2019)
2. “Практическое машинное обучение с помощью Scikit-Learn, Keras и TensorFlow” (2-е издание) Орельен Жерон (2019)
3. **“Нейронные сети и глубокое обучение” (Neural Networks and Deep Learning) / Майкл Нильсен (2019)**
4. “Машинное обучение: методы, модели, применение” Воронцов К.В. (2019)
5. Справочник по Python для обработки данных: основные инструменты для работы с данными» Джейка ВандерПласа (2019)
6. “Разработка функций для машинного обучения” Элис Чжэн и Аманды Казари (2018)
7. “Интеллектуальный анализ данных” Дюк В. (2021)
8. “Распознавание образов и машинное обучение” Кристофера М. Бишопа (2020)
9. Дизайн нейронных сетей (Designing Neural Networks) / Джейсон Браунли.(2020)