# Московский авиационный институт (национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

**Кафедра вычислительной математики и программирования**

## Курсовой проект по курсу «Методы, средства и технологии мультимедиа»

Студент: Кузьмичев А. Н.

Преподаватель: Вишняков Б.В.

Дата: 25.12.2021

Оценка:

Подпись:

**Москва, 2021**

# Ознакомительная часть

# Введение:

## Постановка задачи:

Выбрать задачу (классификация или регрессия), датасет и метрику качества. Вы- бранные данные необходимо визуализировать и проанализировать. После этого вы- полнить препроцессинг. Затем реализовать алгоритм классификации, проверить качество обучения, сравнить с моделью из sklearn.

## Описание данных:

## Данные были получены из изображений, взятых с подлинных и поддельных образцов, похожих на банкноты. Для оцифровки использовалась промышленная камера, обычно используемая для проверки отпечатков. Конечные изображения имеют размер 400х400 пикселей. Благодаря объективу и расстоянию до исследуемого объекта были получены изображения в оттенках серого с разрешением около 660 dpi.

## Информация об атрибутах:

## 1. Дисперсия изображения с вейвлет-преобразованием (непрерывная)

## 2. Асимметрия изображения с вейвлет-преобразованием (непрерывная)

## 3. Эксцесс изображения с вейвлет-преобразованием (непрерывный)

## 4. Энтропия изображения (непрерывная)

## 5. Класс (целое число)

## Вариант:

Логистическая регрессия. Будем предсказывать класс (является ли купюра фейком, или нет).

# Теоретическая часть

# Логистическая регрессия

# Логистическая регрессия - частный случай обобщенной линейной регрессии. Предполагается, что зависимая переменная принимает два значения и имеет биномиальное распределение. На практике логистическая регрессия используется для решения задач классификации с линейно-разделяемыми классами. Логистическая регрессия применяется для прогнозирования вероятности возникновения некоторого события по значениям множества признаков. Для этого вводится зависимая переменная y, принимающая значения 0 и 1 и множество независимых переменных на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной.

# Итак, пусть объекты задаются n числовыми признаками и пространство признаковых описаний в таком случае . Пусть Y — конечное множество меток классов и задана обучающая выборка пар «объект-ответ» .

# Рассмотрим случай двух классов: . В логистической регрессии строится линейный алгоритм классификации вида

# где – вес j-го признака, – порог принятия решения, – вектор весов, - скалярное произведение признакового описания объекта на вектор весов. Предполагается, что искусственно введён нулевой признак: .

# Задача обучения линейного классификатора заключается в том, чтобы по выборке настроить вектор весов w.  В логистической регрессии для этого решается задача минимизации эмпирического риска с функцией потерь специального вида:

# После того, как решение w найдено, становится возможным не только вычислять классификацию для произвольного объекта x, но и оценивать апостериорные вероятности его принадлежности классам:

# где - сигмоидная функция.

# Практическая часть

# Предобработка данных

# Загрузим датасет

# 

# Взглянем на размер данных и наличие пустых элементов

# 

# А затем – на корреляцию между особенностями

# 

# Исключим первый столбец, так как он имеет высокую корреляцию с двумя другими

# 

# Разобьем выборку на тестовую и обучающую

# 

# Реализация алгоритма

class LogisticRegression:

    def \_\_init\_\_(self, \*, reg\_param=1.0, lr=0.01, max\_iter=100):

        self.\_a  = reg\_param

        self.\_lr = lr

        self.\_max\_iter = max\_iter

        self.\_feat\_count = 0

        self.\_weights = None

        self.\_bias = 0

    def fit(self, feats, labels):

        if self.\_weights is None:

            self.\_feat\_count = feats.shape[1]

            self.\_weights = np.zeros(self.\_feat\_count)

        elif self.\_feat\_count != feats.shape[1]:

            err = f"Feature count does not match previous count {self.\_feat\_count}"

            raise ValueError(err)

        costs      = np.zeros((self.\_max\_iter))

        precisions = np.zeros((self.\_max\_iter))

        recalls    = np.zeros((self.\_max\_iter))

        for i in range(self.\_max\_iter):

            pred = self.\_predict(feats)

            costs[i] = self.\_cost(pred, labels)

            precisions[i] = precision(pred.round().astype(int), labels)

            recalls[i] = recall(pred.round().astype(int), labels)

            sample\_count = len(labels)

            norm\_coef = self.\_a

            coef = np.mean(pred - labels)

            dw = np.dot(feats.T, pred - labels) / sample\_count + norm\_coef \* self.\_weights

            db = coef + norm\_coef \* self.\_bias

            self.\_weights -= dw \* self.\_lr

            self.\_bias    -= db \* self.\_lr

        return {

            "cost": costs,

            "precision": precisions,

            "recall": recalls,

            "max\_iter": self.\_max\_iter,

        }

    def predict(self, X):

        return self.\_predict(X).round()

    def \_predict(self, feats):

        weights = self.\_weights

        bias    = self.\_bias

        return 1 / (1 + np.exp(-( np.dot(feats, weights) + bias)))

    def \_cost(self, preds, labels):

        return -1 \* np.mean(

            np.multiply(labels, np.log(preds)) \

            + np.multiply(1 - labels, np.log(1 - preds))

        )

# Метрики

При обучении будут использоваться стандартные метрики:

* **Precision** (точность) - количество правильно классифицированных положительных предметов из выбранных для классификации:

precision=

* **Recall** (полнота) - количество правильно классифицированных положительных предметов из всех возможных (т.е. множество правильно и неправильно классифицированных предметов):

recall=

TP - количество корректно классифицированных положительных предметов, FP - количество некорректно классифицированных отрицательных предметов, TP - количество некорректно классифицированных положительных предметов.

Можно также использовать **accuracy** (количество правильно классифицированных предметов из всех доступных), но её точность малозначима, так как классов неравное количество.

# Результаты

# 

|  |  |  |  |
| --- | --- | --- | --- |
|  | Алгоритм SKL | Алгоритм 1 (4 лр) | Алгоритм 2 (КП) |
| Recall | 0.9015544041450777 | 0.8258426966292135 | 0.8305084745762712 |

# Для оптимизации алгоритма обнулил reg\_param, тем самым убрав L2 нормализацию.