Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Теоретическая информатика и компьютерные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ ПО КУРСУ ЧИСЛЕННЫЕ МЕТОДЫ:

Численный анализ данных о разрушающей силе торнадо

Студент _____

_____

подпись, дата *фамилия, и.о.*

Научный руководитель _____

_____

подпись, дата *фамилия, и.о.*

2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире торнадо, как стихийное бедствие, представляет серьезную угрозу для жизни и имущества людей. Проблема увеличения случаев возникновения торнадо становится все более значимой ввиду изменений климата. Поэтому разработка точных и надежных методов прогнозирования является особенно актуальной задачей для безопасности общества.

Использование алгоритмов, основанных на численных методах статистического анализа данных, позволяет анализировать большие объемы информации, в том числе метеорологическую информацию и признаки окружающей местности. Благодаря этому применение численных методов демонстрирует потенциал в эффективном решении задачи прогнозирования силы торнадо.

Целью данной курсовой работы является изучение и сравнение актуальных численных методов статистического анализа данных, а также создание предсказывающих моделей для прогнозирования силы торнадо. В результате работы предполагается выявить наиболее подходящую и точную модель для данной задачи и реализовать приложение, предсказывающее разрушающую силу торнадо по характеристикам торнадо.

1. Обзор предметной области

1.1 Классификация торнадо

Для оценки интенсивности и разрушающей силы торнадо была создана так называемая шкала Фудзиты. Это шкала, которая отвечает за классификацию торнадо по степени серьезности на основе ущерба, который они могут причинить. Данный метод классификации был предложен в 1971 году профессором Теодором Фудзитой [1].

Однако многочисленные климатические исследования показали, что шкала Фудзиты (F-шкала) переоценивает скорости ветра в категориях сильного (F2-F3) и разрушительного (F4-F5) урона. Поэтому была предложена улучшенная версия этой шкалы, которая была создана Национальной метеорологической службой США [2]. Расширенная шкала Фудзиты (EF-шкала) имеет шесть категорий интенсивности силы торнадо от нуля до пяти, представляющих возрастающие степени ущерба (см. таблица 1). EF-шкала позволяет более тщательно отразить данные о повреждениях от торнадо и точно сопоставить скорость ветра с сопутствующим ущербом от опасного природного явления.

Таблица 1 - Категории расширенной шкалы Фудзиты

Категория	**Оценка скорости ветра**			**Потенциальный урон**
	м/ч	**км/ч**	**м/с**	
EF0	65 - 85	105 - 137	29 - 37	Слабый урон. Повреждает дымовые трубы, дорожные знаки и телевизионные вышки, ломает старые деревья, сносит вывески, разбивает окна.
EF1	86 - 110	138 - 177	38 - 49	Умеренный урон. Срывает крышу с домов, сильно повреждает и/или опрокидывает мобильные и деревянные дома, выбивает окна, перемещает автомобили, вырывает большие деревья с корнем, разрушает лёгкие постройки и гаражи.

EF2	111 - 135	178 - 217	50 - 61	Значительный урон. Срывает крыши с хорошо построенных домов, разрушает мобильные и деревянные дома, вырывает деревья с корнем, сдувает автомобили.
EF3	136 - 165	218 - 266	62 - 74	Сильный урон. Срывает крыши и разрушает стены хорошо построенных домов, опрокидывает поезда, разрушает лёгкие дома, вырывает многие деревья с корнем, поднимает автомобили в воздух, искривляет небоскрёбы, срывает лёгкое покрытие с дороги.
EF4	166 - 200	267 - 322	75 - 89	Разрушительный урон. Разрушает хорошо построенные дома, поднимает в воздух лёгкие дома, может разрушить небоскрёбы, переносит большие деревья на некоторое расстояние, переносит автомобили на некоторое расстояние.
EF5	>200	>322	>90	Невероятный урон. Срывает с фундамента и разрушает хорошо построенные дома, переносит автомобили на расстояние более 100 метров, вырывает с корнем деревья и нередко срывает с них кору, срывает асфальт, может разрушить мосты, разрушает небоскрёбы, повреждает стальные железобетонные конструкции и высокие здания.

Диаграмма частоты появлений торнадо разных категорий представлена на рисунке 1. Самыми частыми являются торнадо категорий EF0 И EF1, их частота составляет 52.82% и 32.98%, а самым редким явлением – торнадо категории EF5 с частотой 0.05%.

Рисунок 1 - Частота появлений торнадо в процентах

1.2 Численные методы статистического анализа данных

В рамках данной курсовой работы по прогнозированию разрушающей силы торнадо был проведен сравнительный анализ нескольких методов статистического анализа данных, представляющих собой стохастические модели. У каждой из моделей имеются свои методы решения задачи классификации и уникальные особенности, что позволяет им быть эффективно применяемыми в разнообразных ситуациях анализа данных. Далее произведен обзор моделей, используемых в работе.

1.2.1 Логистическая регрессия

Логистическая регрессия является популярной статистической моделью для бинарной классификации, но несмотря на это, ее также можно использовать для классификации нескольких классов, то есть полиномиальной классификации, что и необходимо в контексте поставленной задачи. Для этого необходимо рассмотреть не одну модель, а k моделей, каждая из которых будет выполнять бинарную классификацию, отличая k-й класс от остальных.

Рассмотрим механизм модели логистической регрессии. Модель стремится вычислить вероятность принадлежности к определенному классу. Однако вероятность , по определению, величина от 0 до 1, поэтому для решения задачи и соблюдения данного ограничения используется преобразование данных предсказания с помощью функции логит-преобразования: $P = \frac{1}{1+e^{-<w_i x_i>}}$, где P-вероятность интересующего события, а $z =< w_i x_i > = \sum_i w_i x_i + b$ – значение, представляющие собой линейную комбинацию весов w_i, входных значений x_i и некоторого смещения b. Функцию, записанную в правой части рассматриваемого выражения, также называют сигмоидой. Она обозначается $\sigma(z) = \frac{1}{1+e^{-z}}$ и является функцией активации в логистической регрессии [3].

После получения результата преобразования входного значения через сигмоидальную функцию используется алгоритм градиентного спуска для оптимизации модели и минимизации ошибки по формуле:

$$w_{i+1} = w_i - learning\_rate * \nabla L(f(x; w), y),$$

где w_{i+1} – обновленное значение весов,

w_i – предыдущее значение весов,

$learning\_rate$ – скорость обучения,

$\nabla L(f(x; w), y)$ – градиент функции потерь для весов.

Функция потерь вычисляется по формуле:

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\widehat{y_i}) + (1 - y_i)\log(1 - \widehat{y_i}),$$

где y_i – результат применения функции активации,

$\widehat{y_i}$ – реальный результат,

n – число наблюдений.

Градиент для весов вычисляется как:

$$\frac{\partial L(f(x;w),y)}{\partial w} = \frac{1}{n}(\hat{y} - y)x^T,$$

где x^T – транспонированный вектор входных данных.

1.2.2 Дерево решений

Классификаторы на основе дерева решений относятся к одним из наиболее эффективных средств анализа данных и предиктивной аналитики, позволяющих решать задачи классификации.

Процесс построения деревьев решений связан с последовательным, рекурсивным разделением обучающего набора данных на подмножества с использованием решающих правил в узлах. Моделирование процесса принятия решений осуществляется через иерархическую структуру ветвлений, где каждый узел представляет собой проверку атрибута, каждая ветвь обозначает результат проверки, а каждый лист содержит результат классификации.

Рекурсивный процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями. Объявление узла листом происходит либо когда он содержит единственный объект (или объекты одного класса), либо по достижении некоторого условия остановки, например, максимальной глубины дерева. Ограничение на глубину дерева часто используется для контроля переобучения, особенно в случаях, когда стандартное дерево решений может иметь тенденцию к переобучению на тренировочных данных.

Ключевыми формулами, которые используются в методе дерева решений являются:

Критерий примеси Джини для выбора разделения:

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2,$$

где D – подмножество данных,

c – количество классов,

p_i – доля объектов класса i в подмножестве D.

Этот критерий позволяет определить, какой атрибут наилучшим образом разделяет данные на классы, минимизируя неопределенность в каждой подгруппе. Для разделения узла выбирается атрибут с наименьшим значением примеси Джини [4].

Функция разделения обучающего набора D на подмножества по атрибуту A:

$$Gini_A(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2),$$

где D – подмножество данных размерности n, которое разбивается на множества D_1 и D_2 с размерностями n_1 и n_2 соответственно,

A – атрибут, выбранный для разделения узла дерева.

1.2.3 Метод K-ближайших соседей

Метод K-ближайших соседей является одним из наиболее интуитивно понятных алгоритмов статистического анализа данных, используемых для классификации. Этот метод не требует явного обучения модели, что делает его уникальным среди алгоритмов классификации.

Метод основан на идее анализа близости между объектом и уже классифицированными точками в обучающем наборе данных, что и обеспечивает эффективное прогнозирование категории для новых данных [3].

Алгоритм данного метода состоит из нескольких шагов:

1) Определение k — числа ближайших соседей. Это важный параметр, определяющий количество ближайших объектов в обучающем наборе данных, которые будет использоваться для классификации нового объекта.

2) Для каждого объекта в обучающем наборе выполняется вычисление расстояния до нового объекта. В подавляющем большинстве случаев обычное евклидово расстояние:

$$\rho(x, x') = \sqrt{\sum_i (x_i - x'_i)^2},$$

где x_i – признак нового объекта, а x'_i– признаки объекта из обучающего набора.

3) Сортировка объектов после вычисления расстояний и выбор k объектов с наименьшим расстоянием. Классификация нового объекта осуществляется на основе большинства голосов среди этих ближайших соседей.
4) Для каждого класса выполняется расчет вероятности как доля соседей этого класса среди k ближайших.

1.2.4 Линейный дискриминантный анализ

Линейный дискриминантный анализ (ЛДА) – это метод статистического анализа, часто применяемый в машинном обучении для поиска линейных комбинаций входных признаков, которые наилучшим образом решают задачи дискриминации (различения) объектов наблюдения по набору признаков. То есть метод позволяет изучать различия между двумя или более классами. Метод часто используется для прогнозирования принадлежности объектов к определенным классам на основе значений входных признаков.

Основной целью дискриминации является поиск линейной комбинации признаков (называемых дискриминантными признаками), которые позволили бы наилучшим образом разделить рассматриваемые группы, создать разделяющие гиперплоскости для классификации. Функция линейной комбинации выглядит так:

$$d_{km} = a_0 + a_1 x_{1km} + \cdots + a_p x_{pkm},$$

где d_{km} – значение дискриминирующей функции для m-го наблюдения k-й группы, $m = 1, \dots, n; k = 1, \dots, g$,

x_{ikm} – значение дискриминантного признака для m-го наблюдения k-й группы,

p – число дискриминантных признаков.

При помощи ЛДА мы можем ранжировать дискриминантные функции по их способности разделять классы и выбирать наиболее информативные для классификации.

Важно отметить, что хотя метод и работает с информацией, которая определяет принадлежность объекта к одному из классов, но сам по себе классификатором не является, а используется как часть линейной классификационной модели.

1.2.5 Наивный байесовский классификатор

Наивный байесовский классификатор – это метод классификации, основанный на применении теоремы Байеса с допущением о независимости признаков. Этот метод широко используется в машинном обучении для оценки вероятности принадлежности объекта к определенной категории на основе значений его признаков.

Классификатор применяет теорему Байеса для оценки вероятности принадлежности объекта к определенной категории при условии известных значений признаков.

Рассмотрим механизм работы. Пусть каждый пример x принимает значения из множества V и описывается атрибутами $< a_1, a_2, \dots a_n >$. Необходимо найти наиболее вероятное значение данного атрибута $v \in V$, т.е.

$$v_{MAP} = argmax_{v \in V} P(x = v \mid a_1, \dots, a_n).$$

Теорема Байеса говорит о том, что:

$$P(x = v \mid a_1, \dots, a_n) = \frac{P(\ a_1, \dots, a_n | x = v) P(x = v\ )}{P(a_1, \dots, a_n)}.$$

Тогда имеем:

$$v_{MAP} = argmax_{v \in V} P(x = v \mid a_1, \dots, a_n) = \frac{P(\ a_1, \dots, a_n | x = v) P(x = v\ )}{P(a_1, \dots, a_n)} =$$

$$= argmax_{v \in V} P(\ a_1, \dots, a_n | x = v) P(x = v\ )$$

Наивный байесовский классификатор учитывает априорные вероятности принадлежности объектов к классам, что позволяет учесть априорное знание о распределении классов.

2. Проектирование и разработка

2.1 Описание набора данных

В первую очередь, прежде чем строить классификационную модель, необходимо проанализировать наши данные, чтобы понять с чем мы имеем дело. В данной работе был использован набор данных с Kaggle [5].

Датасет по торнадо представляет собой большой набор данных, включающий различные параметры данного природного явления, зарегистрированного в США в различных штатах в период с 1950 по 2022 год. Датасет предоставляет информацию о точных датах, размерах и геопозиции торнадо, а также данные о нанесенном ущербе, количестве жертв и раненных вследствие этого стихийного бедствия в течение семи десятилетий. Такое разнообразие данных позволяет провести различные анализы, посмотрев на динамику количества торнадо в разные годы и изучив влияние торнадо на число пострадавших и материальный ущерб.

В рассматриваемом наборе представлены следующие атрибуты:

Номер торнадо (om) – порядковый номер торнадо в данном году.

Год (yr), месяц (mo) и день (dy) торнадо. Год принадлежит диапазону от 1950 до 2022. Месяц – от 1 до 12. День месяца – от 1 до 31.

Дата торнадо (date). Дата отображается в формате ISO 8601 ГГГГ-ММ-ДД.

Время торнадо (time). Время отображается как ЧЧ:ММ:СС, где ЧЧ – час, ММ – минуты, а СС – секунды.

Часовой пояс (tz) – это атрибут, представляющий канонический часовой пояс базы данных tz. База данных tz представляет собой информации о часовых поясах мира, предназначенную в первую очередь для использования с компьютерными программами и операционными системами.

Дата и время в формате UTC (datetime_utc) – атрибут даты и времени, нормализованный по UTC (всемирное координированное время).

Штат (st) и номер штата (stf). Штат представлен двухбуквенным почтовым сокращением, а номер является федеральным номером штата по федеральному стандарту по обработке информации FIPS.

Разрушающая сила торнадо, магнитуда (mag). Сила определяется по шкале EF.

Количество раненых (inj) и погибших (fat) – это целочисленные атрибуты, указывающие на количество потерпевших и жертв торнадо.

Ущерб торнадо (loss) – это атрибут, содержащий оценку ущерба в долларах.

Географическая широта (slat) и долгота (slon) начальной точки торнадо. Данные атрибуты определяют точку появления торнадо в десятичных градусах.

Географическая широта (elat) и долгота (elon) заключительной точки торнадо. Данные атрибуты определяют заключительную точку торнадо в десятичных градусах.

Длина (len) и широта (wid) торнадо, измеряемые в милях и ярдах соответственно.

Количество затронутых штатов (ns) – количество штатов, затронутых данным торнадо. Может быть равно 1, 2 или 3.

Номер штата для данного торнадо (sn). Значение 1 указывает на то, что строка содержит всю информацию о пути этого торнадо по этому штату, 0 означает, что существует как минимум еще одна запись для этого штата для этого торнадо (**om + yr**).

Коды затронутых торнадо штатов для первого (f1), второго (f2), третьего (f3) и четвертого (f4) округов. Коды представлены по стандарту FIPS.

Наличие оценки магнитуды торнадо (fc). Логический атрибут указывает на то, проводилась ли оценка данного торнадо в Национальной службе погоды США (NOAA`s National Weather Service).

Видим, что исходные данные являются необработанными, «сырыми», поэтому они требуют предварительной очистки и обработки, чтобы быть готовыми к анализу и использованию.

2.2 Предобработка данных

Очистка и предварительная обработка данных являются критически важными этапами в работе с любым большим набором данных и требует тщательного подхода, поскольку точность анализа напрямую зависит от подготовки начальной информации, а ошибки на этом этапе могут привести к некорректным выводам при последующем моделировании.

В рамках процесса предобработки данных датасета были применены следующие техники:

Очистка данных. При выполнении очистки датасета были обнаружены и удалены дубликаты записей. Также была выполнена проверка на наличие пропущенных значений, аномально высоких или низких значений, которые могут указывать на ошибки ввода или измерения, однако таких записей не было обнаружено.

Нормализация и масштабирование данных. Этот шаг заключается в приведении всех числовых переменных к единому масштабу, что предотвращает доминирование переменных с более высокими значениями в процессе моделирования. Для этого использовался метод минимаксной нормализации.

Разделение данных. Очищенные и нормализованные данные были разделены на обучающую и тестовую выборки для будущей оценки производительности моделей машинного обучения. Чаще всего используется разделение в соотношении 70% и 30% для обучения и тестирования соответственно, однако это соотношение может меняться в зависимости от размера и специфики датасета. В данном случае использовалось соотношение 75% на 25%

2.3 Анализ набора данных

Исследовательский анализ данных представляет собой ключевой этап в обработке и изучении информации, содержащейся в датасете. Этот процесс включает в себя визуализацию данных, выбор входных признаков модели, а также поиск и анализ корреляций параметров с целевым признаком.

Целевым признаком модели в данном случае является разрушающая сила торнадо (магнитуда). Магнитуда торнадо определяется по расширенной шкале Фудзиты, которая определяет принадлежность торнадо к определенной категории: EF0, EF1, EF2, EF3, EF4, EF5. Разделение на категории по разрушающей силе позволяет оценить уровень опасности и возможного воздействия этого стихийного бедствия.

Целью данного исследовательского анализа является выявление характеристик торнадо, которые влияют на магнитуду торнадо и ее рост. Такие признаки будут являться входными для предсказывающей модели. Также необходимо изучить опасность урона, предоставляемого торнадо в разные годы, проверить гипотезу о линейной зависимости между разрушающей силой торнадо и количеством жертв и определить тенденцию частоты появлений торнадо в ближайшие годы.

Всего в наборе данных присутствует 27 колонок, из них были выбраны колонки, которые потенциально могут быть полезны в дальнейшем анализе и иметь зависимости между собой:

- yr – численный атрибут, отражающий год зарегистрированного торнадо (1950-2022),
- mo – численный атрибут, отражающий месяц зарегистрированного торнадо (1-12),
- st – категориальный атрибут, показывающий штат зарегистрированного торнадо,

- mag – целевой признак модели и категориальный атрибут, показывающий магнитуду (разрушающую силу) торнадо по расширенной шкале Фудзиты,
- inj – численный атрибут, отражающий количество травм вследствие торнадо,
- fat – численный атрибут, отражающий количество погибших вследствие торнадо,
- loss – численный атрибут, предоставляющий информацию о предполагаемой потере имущества в долларах,
- len – численный атрибут, отражающий длину торнадо в милях,
- wid – численный атрибут, отражающий ширину торнадо в ярдах,
- slat – численный атрибут, показывающий начальную географическую широту торнадо в градусах,
- slon – численный атрибут, показывающий начальную географическую долготу торнадо в градусах.

Все рассмотренные колонки являются числовыми за исключением колонки штата, однако штаты США кодируются двухбуквенной почтовой аббревиатурой, что было решено оставить, не конвертируя в численный тип. Поскольку для начала необходимо выделить признаки, используемые в модели, а потом совершать перевод в другой тип данных по необходимости.

Остальные колонки не являются полезными для дальнейшего использования в анализе данных.

Для изучения распределения числовых переменных и выявления возможных зависимостей была проведена визуализация данных в виде гистограмм, диаграмм и точечных графиков, представленных на рисунках ниже.

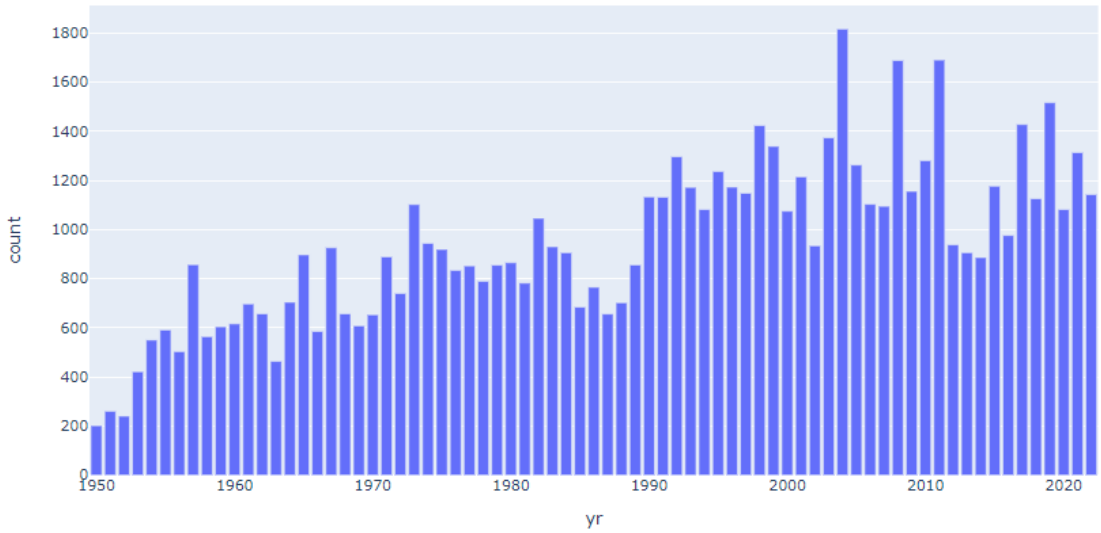

Рисунок 2 - Распределение торнадо по годам

На рисунке 2 было рассмотрено распределение появлений торнадо в разные годы промежутка 1950-2022 гг. Можно заметить тенденцию на увеличение общего числа торнадо с каждым годом. Проверим это, построив прогноз, где на оси x откладываются значения года, на оси y - количество событий в этот год, и добавляется линия тренда для визуализации общей тенденции (см. рисунок 3).

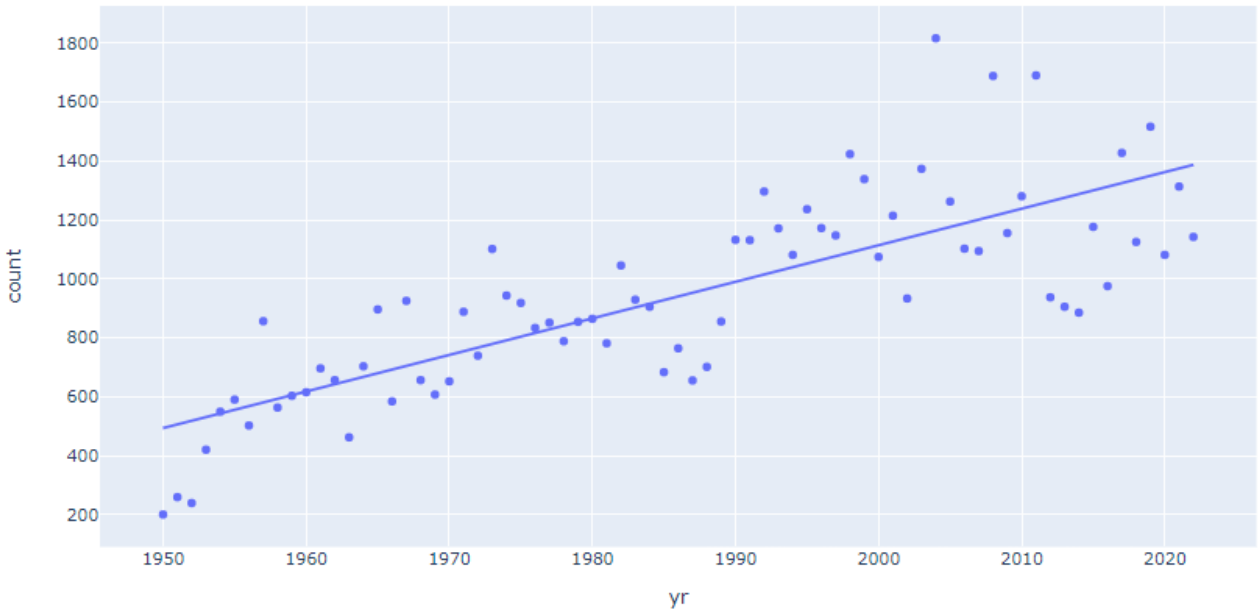

Рисунок 3 - Диаграмма рассеяния с линией тренда

Действительно, виден тренд на увеличение числа торнадо.

Однако, анализ разрушающей силы торнадо, представленный на рисунках 4-5 показывает, что большинство торнадо относятся к категории EF0 (слабый урон) по усовершенствованной шкале Фудзиты.

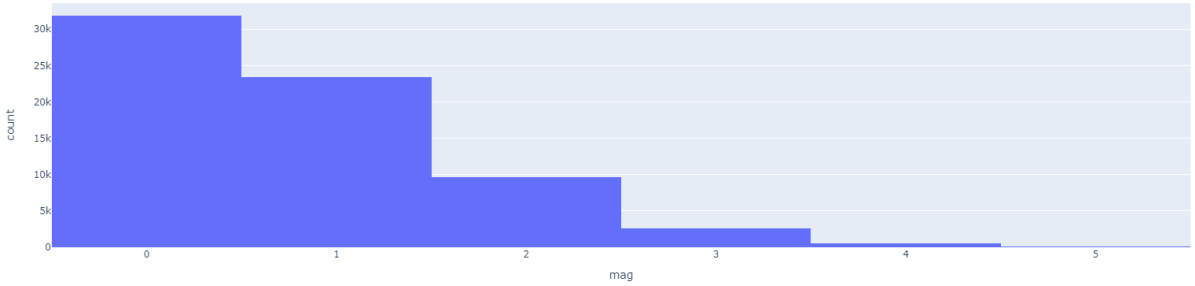

Рисунок 4 - Распределение магнитуды торнадо по EF-шкале

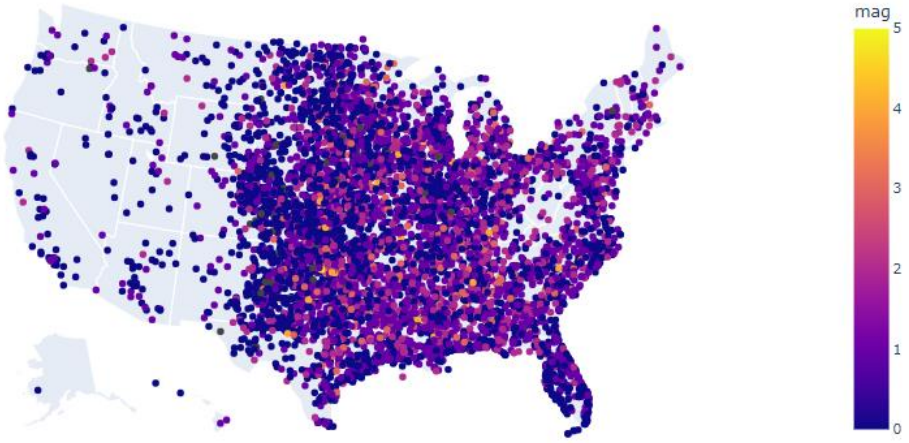

Рисунок 5 - Распределение магнитуды торнадо по EF-шкале на карте

Теперь посмотрим на распределение людей, получивших травмы и число погибших и попробуем проанализировать.

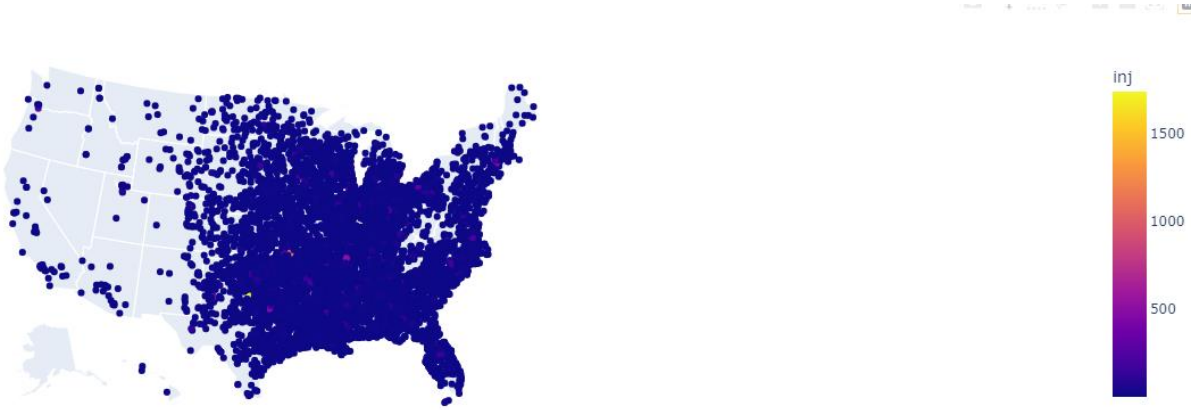

Рисунок 6 - Распределение получивших травмы вследствие торнадо

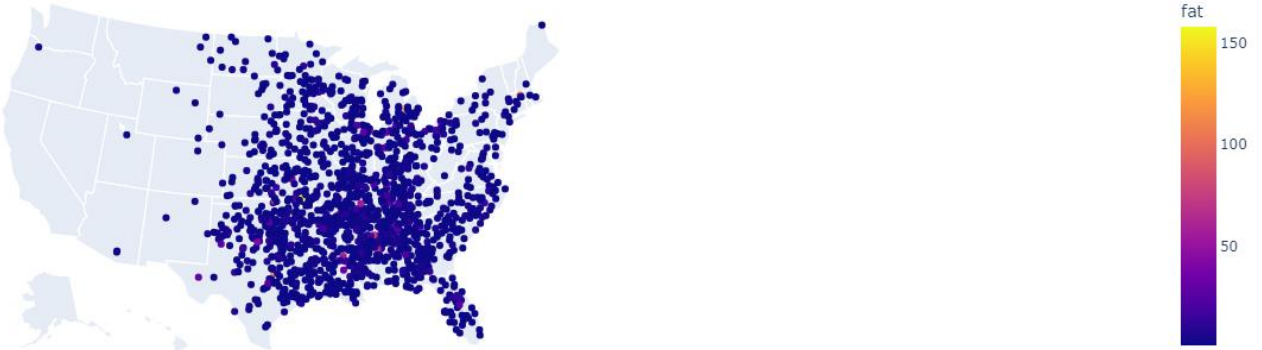

Рисунок 7 - Распределение погибших вследствие торнадо

На рисунках 6-7 видно, что подавляющее число получивших травмы небольшое, а число смертельных случаев на порядок меньше. Более того по рисункам карт можно заметить, что большинство торнадо образуется чаще всего на восточной части страны.

Далее был проведен анализ с целью поиска штата с наибольшей частотой появления торнадо. Результаты представлены на рисунках 8-9.

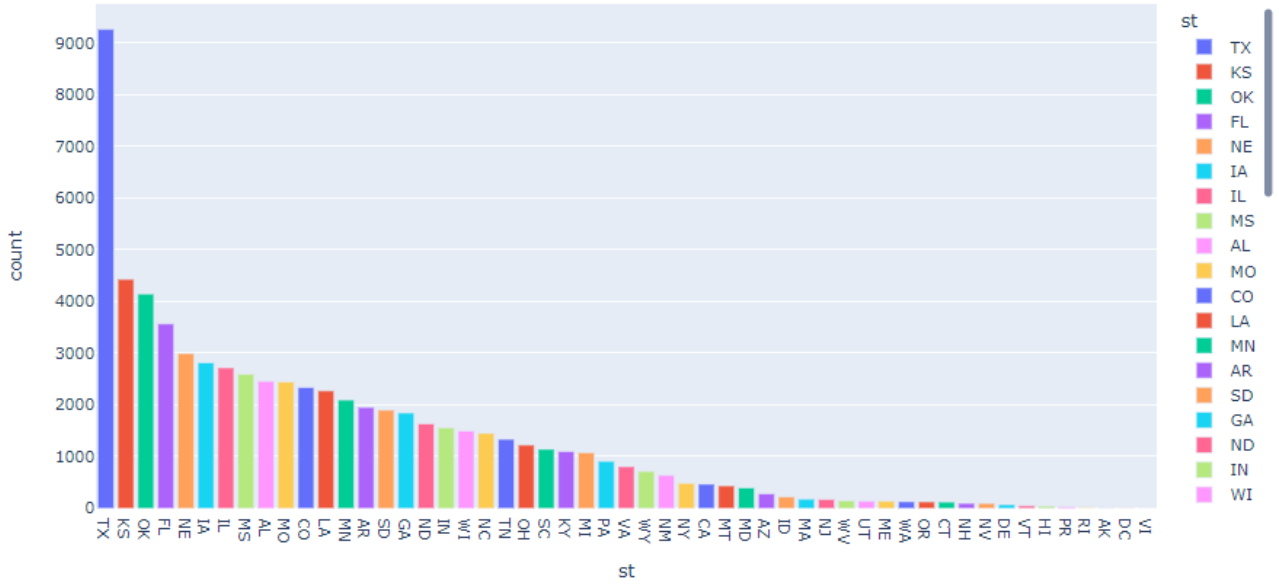

Рисунок 8 - Зависимость общего числа торнадо в период 1950-2022 от штата

Рисунок 9 - Карта распределения общего числа торнадо в период 1950-2022 по штатам

Действительно, наибольшая часть торнадо приходится на восток страны, более того чаще всего торнадо встречаются на юге страны – в штате Техас.

Далее был проведен анализ и поиск корреляций между данными. Однако, при работе с категориальными и количественными признаками коэффициент корреляции не является подходящим методом для оценки их взаимосвязи,

поскольку обычно используется для оценки линейной взаимосвязи между двумя количественными непрерывными переменными.

Для оценки связи между категориальным признаком и количественными признаками часто используются различные методы визуализации, например графики рассеяния. Такие графики, а также результаты анализа корреляций между разрушающей силой торнадо по EF-шкале и количественными характеристиками торнадо (длина, ширина, географические координаты, количество смертей и т.д.) представлены на рисунках ниже.

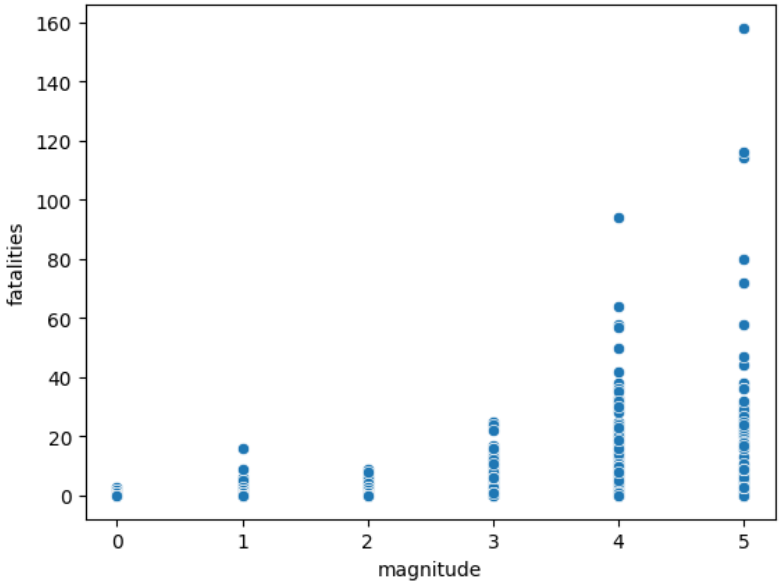

Рисунок 10 - Корреляция между разрушающей силой торнадо и числом погибших

Далее для упрощения анализа и балансировки данных, они были разделены не на 6 категорий по шкале Фудзиты, а на 3категории: только EF0, только EF1, диапазон EF2-EF5.

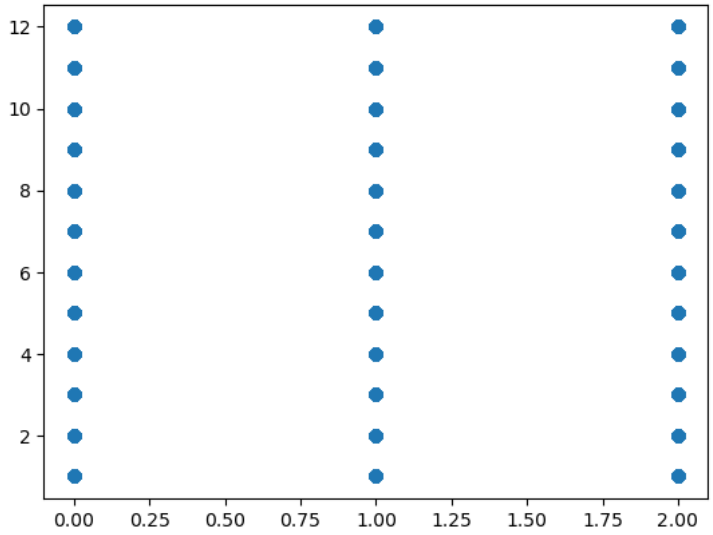

Рисунок 11 - Отсутствие корреляции между разрушающей силой торнадо и месяцем появления

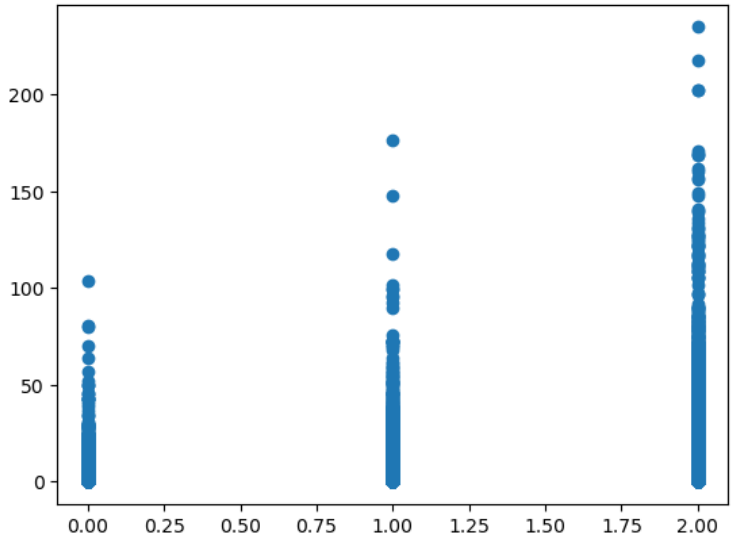

Рисунок 12 - Корреляция между разрушающей силой торнадо и длиной торнадо

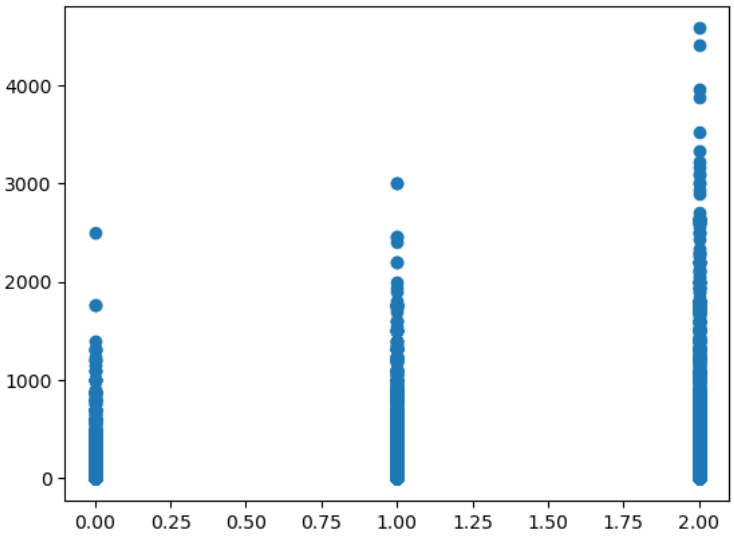

Рисунок 13 - Корреляция между разрушающей силой торнадо и шириной торнадо

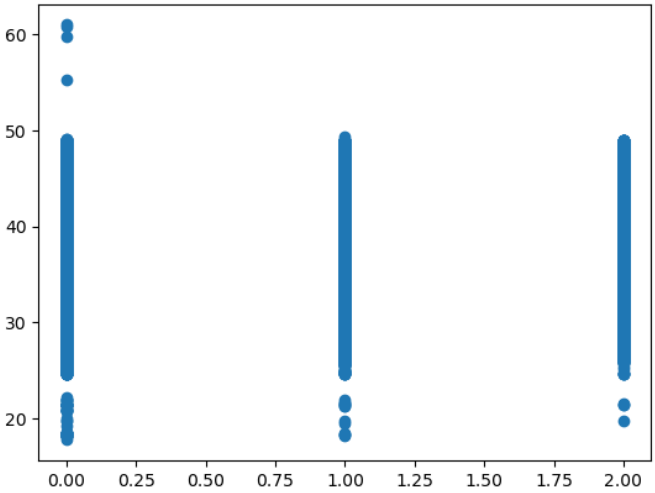

Рисунок 14 - Слабая корреляция между разрушающей силой торнадо и начальной географической широтой. Негативный тренд

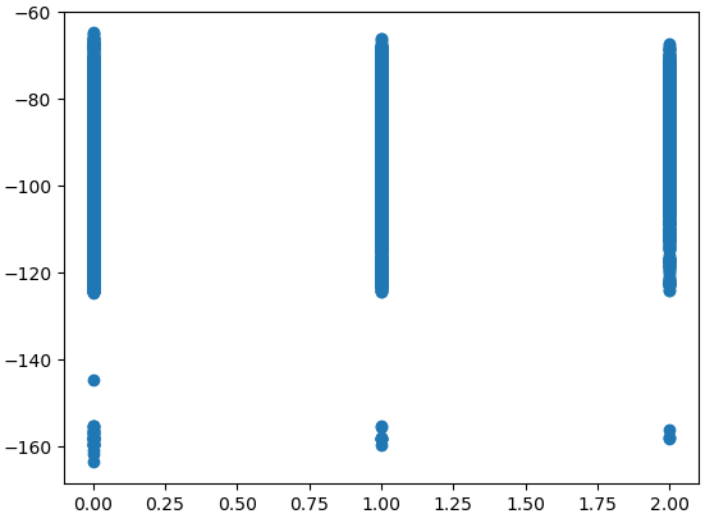

Рисунок 15 - Слабая корреляция между разрушающей силой торнадо и начальной географической долготой. Негативный тренд

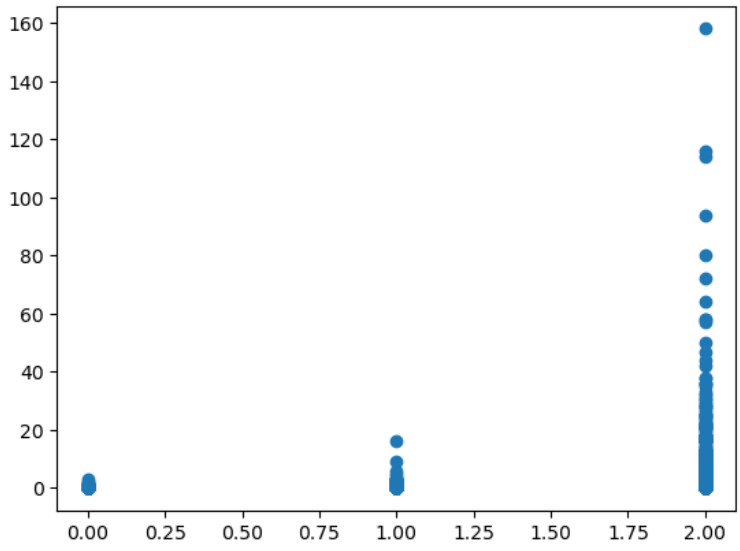

Рисунок 16 - Корреляция между разрушающей силой торнадо и числом погибших

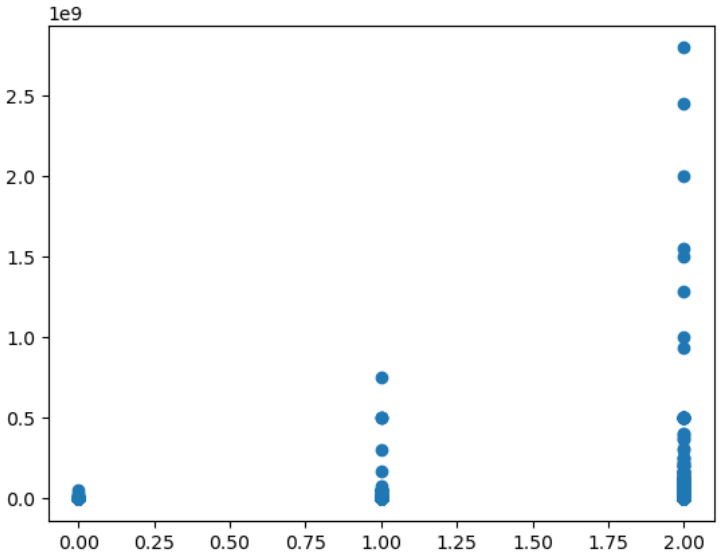

Рисунок 17 - Корреляция между разрушающей силой торнадо и предполагаемым материальным ущербом

Анализ корреляций между различными параметрами и разрушающей силой торнадо показали, какие параметры имеют влияние на рост разрушающей силы, вследствие чего были выбраны признаки, подаваемые на вход предсказывающей модели.

Таким образом, предсказывающая модель получает на входе следующие признаки:

- длина торнадо в милях,
- ширина торнадо в ярдах,
- количество людей, погибших вследствие торнадо,
- географические широта и долгота места происшествия в градусах.

2.4 Требования к ПО

Было решено разработать приложение, имеющее клиент-серверную архитектуру.

Сервер должен получать запросы пользователя с информацией о таких признаках торнадо, как длина и ширина явления, а также штат, где было замечено торнадо. Далее сервер выполняет геокодирование полученных данных, определяя

широту и долготу населенного пункта, форматирует данные и передает их в предсказывающую модель для получения прогноза разрушающей силы торнадо. После того, как модель выполнила работу, сервер передает ответ на клиентскую сторону.

Клиент представляет собой пользовательский интерфейс, через который происходит взаимодействие пользователя с моделью статистического анализа. Пользователь приложения вводит данные о торнадо, отправляет данные и получает ответ. Также было решено добавить визуализацию карты страны для наглядного представления информации о появлении торнадо в рассматриваемый период. В процессе интерактивного взаимодействия пользователя с интерфейсом, улучшается восприятие информации и развивается положительный пользовательский опыт.

В соответствии с поставленными требованиями в пользовательском интерфейсе должны быть предусмотрены:

- форма для ввода информации от пользователя,
- кнопка для отправления запроса на сервер,
- поле для вывода предсказанной категории разрушающей силы торнадо,
- интерактивная карта, отображающая зарегистрированные торнадо в разных штатах страны.

3. Программная реализация

Для реализации приложения были использованы следующие технологии:

- язык программирования Python,
- фреймворк *Flask*. Это легкий, гибкий и простой в использовании веб-фреймворк для Python, который позволяет создавать веб-приложения [6],
- *Plotly* и *Matplotlib.pyplot* – библиотеки, позволяющие визуализировать данные, строить разнообразные типы графиков, в том числе интерактивные,
- *Pandas* – библиотека Python, предназначенная для обработки и анализа данных. Она предоставляет инструменты для работы с табличными данными, а также мощные средства для преобразования, фильтрации и агрегации данных [7],
- *Scikit-learn (Sklearn)* – библиотека машинного обучения для Python, которая предоставляет широкий спектр алгоритмов машинного обучения, включая методы классификации, регрессии, кластеризации, а также инструменты для оценки и выбора моделей [8],
- *Pickle* – модуль Python, предназначенный для сериализации и десериализации объектов. Pickle также используется для сохранения моделей машинного обучения [9].
- HTML, CSS, JavaScript для создания пользовательского интерфейса.

На рисунках 18-19 представлен код для запуска приложения, созданного с использованием фреймворка *Flask* с поддержкой асинхронных запросов, что позволяет отправлять и получать запросы в фоновом режиме без перезагрузки страницы. С помощью модуля *Pickle* загружается объект лучшей модели из рассматриваемых предсказывающих моделей, а далее производится обработка POST-запроса от пользователя, преобразование данных для входа в модель с использованием метода масштабирования данных ***MinMaxScaler***. После выполнения функции **predict()** результат предсказания сопоставляется с категорией EF-шкалы и возвращается на сервер.

```python
from flask import Flask, request
from flask_cors import CORS
import pandas as pd
import pickle
from sklearn.preprocessing import MinMaxScaler
from geopy.geocoders import Nominatim

app = Flask(__name__)
CORS(app)

model = pickle.load(open('model.pkl', 'rb'))
tornado_df = pd.read_csv('cleaned.csv')
```

Рисунок 18 - Создание приложения с использованием Flask и загрузка данных и предсказывающей модели

```python
@app.route("/predict", methods=["POST"])
def predict():
    if request.method == "POST":
        leng = float(request.form["leng"])
        wid = float(request.form["wid"])
        fat = float(request.form["fat"])
        place = request.form["place"]
        locator = Nominatim(user_agent="myAppGeocoder")
        location = locator.geocode(place)
        slat = location.latitude
        slon = location.longitude

        data = [[fat, leng, wid, slat, slon]]
        user_df = pd.DataFrame(data, columns=["fat", "len", "wid", "slat", "slon"])
        features = tornado_df
        complete = pd.concat([features, user_df]).reset_index(drop=True)
        scaler = MinMaxScaler()
        scaled_df = scaler.fit_transform(complete)
        output = model.predict([list(scaled_df[-1])])

        category = ""
        if(output[0] == 0):
            category = "EF 0 - Light damage"
        elif(output[0] == 1):
            category = "EF 1 - Moderate damage"
        elif(output[0] == 2):
            category = "EF 2 - Considerable damage"
        elif(output[0] == 3):
            category = "EF 3 - Severe damage"
        elif(output[0] == 4):
            category = "EF 4 - Devastating damage"
        elif(output[0] == 5):
            category = "EF 5 - Incredible damage"
        return { "classify": category }

if __name__ == "__main__":
    app.run(debug=False)
```

Рисунок 19 - Обработка полученных данных о торнадо и выполнение прогнозирования категории разрушающей силы

Теперь рассмотрим реализацию и сравнение моделей статистического анализа данных, которые использовались для прогнозирования разрушающей силы торнадо, и среди которых выбиралась лучшая для использования в приложении.

На рисунке 20 представлено разделение датасета на тренировочные и тестовые выборки с помощью функции ***train_test_split()***, которая по умолчанию делит данные в соотношение 75% к 25%. Очищенные данные тренировочной выборки сохраняются в файл ***cleaned.csv***.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

train_output = X_train[["fat", "len", "wid", "slat", "slon"]]
train_output.to_csv(r'cleaned.csv',index=False)
```

Рисунок 20 - Подготовка тренировочных и тестовых выборок

Далее было произведено сравнение реализаций моделей с помощью библиотеки *Scikit-learn (Sklearn).* Код представлен на рисунках 21-22, а результаты этого сравнения продемонстрированы в разделе тестирования.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Рисунок 21 – Масштабирование данных перед анализом методов классификации

```
## Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(multi_class = 'ovr')
logreg.fit(X_train_scaled, y_train)
print('Accuracy of Logistic Regression on training data',
logreg.score(X_train_scaled, y_train))
print('Accuracy of Logistic Regression on testing data', logreg.score(X_test_scaled,
y_test))

## Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train_scaled, y_train)
print('Accuracy of Decision Tree on training', dt.score(X_train_scaled, y_train))
print('Accuracy of Decision Tree on testing', dt.score(X_test_scaled, y_test))

# Setting max depth
dt2 = DecisionTreeClassifier(max_depth=5)
dt2.fit(X_train_scaled, y_train)
print('Accuracy of Decision tree on training', dt2.score(X_train_scaled, y_train))
print('Accuracy of Decision tree on testing', dt2.score(X_test_scaled, y_test))

# K-nearest neighbor
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)
print('Accuracy of KNN on training', knn.score(X_train_scaled, y_train))
print('Accuracy of KNN on testing', knn.score(X_test_scaled, y_test))

# Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train_scaled,y_train)
print('Accuracy of Linear Discriminant Analysis on training',
lda.score(X_train_scaled, y_train))
print('Accuracy of Linear Discriminant Analysis on testing', lda.score(X_test_scaled,
y_test))

# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train_scaled, y_train)
print('Accuracy of GNB on training', gnb.score(X_train_scaled, y_train))
print('Accuracy of GNB on testing', gnb.score(X_test_scaled, y_test))
```

Рисунок 22 - Сравнительный анализ методов классификации

По результатам (они представлены в разделе тестированя) сравнения точности данных алгоритмов статистического анализа, была выявлена лучшая модель – модель дерева решений с максимальной глубиной 5.

Теперь представим собственную реализацию данной модели и сравним точности собственной реализации и библиотечной функции. Лучшая из моделей будет сохранена (см. рисунок 26) для использования в итоговом приложении.

Дерево решений и отдельный узел реализованы в виде классов.

```
import numpy as np

class Node:
    def __init__(self, gini, num_samples, class_num_samples, predicted_class):
        self.gini = gini
        self.num_samples = num_samples
        self.class_num_samples = class_num_samples
        self.predicted_class = predicted_class
        self.feature_index = 0
        self.porog = 0
        self.left = None
        self.right = None
```

Рисунок 23 - Реализация узла дерева для метода дерева решений

```
class MyDecisionTree:
    def __init__(self, max_depth=None):
        self.max_depth = max_depth

    def fit(self, X, y):
        self.n_classes_ = len(set(y))
        self.n_features_ = X.shape[1]
        self.tree_ = self._grow_tree(X, y)

    def _gini(self, y):
        m = len(y)
        return 1.0 - sum((np.sum(y == c) / m) ** 2 for c in range(self.n_classes_))

    def _best_split(self, X, y):
        m, n = X.shape
        if m <= 1:
            return None, None

        num_parent = [np.sum(y == c) for c in range(self.n_classes_)]
        best_gini = 1.0 - sum((n / m) ** 2 for n in num_parent)
        best_idx, best_porog = None, None

        for idx in range(self.n_features_):
            porogs, classes = zip(*sorted(zip(X[:, idx], y)))
            num_left = [0] * self.n_classes_
            num_right = num_parent.copy()
            for i in range(1, m):
                c = classes[i - 1]
                num_left[c] += 1
                num_right[c] -= 1
                gini_left = 1.0 - sum((num_left[x] / i) ** 2 for x in
range(self.n_classes_))
                gini_right = 1.0 - sum((num_right[x] / (m - i)) ** 2 for x in
range(self.n_classes_))
                gini = (i * gini_left + (m - i) * gini_right) / m
                if porogs[i] == porogs[i - 1]:
                    continue
                if gini < best_gini:
                    best_gini = gini
                    best_idx = idx
                    best_porog = (porogs[i] + porogs[i - 1]) / 2

        return best_idx, best_porog
```

Рисунок 24 - Реализация метода дерева решений

```
    def _grow_tree(self, X, y, depth=0):
        class_num_samples = [np.sum(y == i) for i in range(self.n_classes_)]
        predicted_class = np.argmax(class_num_samples)
        node = Node(
            gini=self._gini(y),
            num_samples=len(y),
            class_num_samples=class_num_samples,
            predicted_class=predicted_class,
        )

        if depth < self.max_depth:
            idx, porog = self._best_split(X, y)
            if idx is not None:
                indices_left = X[:, idx] < porog
                X_left, y_left = X[indices_left], y[indices_left]
                X_right, y_right = X[~indices_left], y[~indices_left]
                node.feature_index = idx
                node.porog = porog
                node.left = self._grow_tree(X_left, y_left, depth + 1)
                node.right = self._grow_tree(X_right, y_right, depth + 1)
        return node

    def _predict(self, inputs):
        node = self.tree_
        while node.left:
           node = node.left if inputs[node.feature_index] < node.porog else
node.right
        return node.predicted_class

    def predict(self, X):
        return [self._predict(inputs) for inputs in X]
```

Рисунок 25 – Реализация метода дерева решений (продолжение)

```
import pickle
pickle.dump(dt2, open('model.pkl','wb'))
```

Рисунок 26 - Загрузка лучшей модели

4. Тестирование приложения

Цель тестирования заключается в проверке работоспособности основных функций приложения, связанных с выполнением предсказаний моделей и визуализацией данных через пользовательский интерфейс.

Сравнительный анализ точности предсказаний рассмотренных реализаций выбранных численных методов представлен в таблице 2.

Таблица 2 - Сравнительный анализ точности реализаций численных методов

Название модели	Точность модели на тренировочном наборе данных	Точность модели на тестовом наборе данных
Логистическая регрессия	0.67	0.66
Дерево решений	0.99	0.64
Дерево решений с максимальной глубиной 5	0.71	0.70
К ближайших соседей	0.77	0.67
Линейный дискриминантный анализ	0.65	0.65
Наивный байесовский классификатор	0.66	0.67

Видим, что лучшие результаты на тестовой выборке показало дерево решений с заданной максимальной глубиной. Несмотря на то, что алгоритм стандартного дерева решений показал отличные результаты на тренировочном наборе данных, обобщающая способность данной модели оказалась хуже.

Далее было произведено сравнение собственной реализации дерева решений с максимальной глубиной с библиотечным алгоритмом.

Название модели	Точность модели на	Точность модели на

	тренировочном наборе данных	тестовом наборе данных
Собственная реализация Дерева решений с максимальной глубиной 5	0.67	0.63
Дерево решений с максимальной глубиной 5	0.71	0.70

К сожалению, собственная реализация метода классификации оказалась хуже и по предсказывающей, и по обобщающей способности. Поэтому выбор пал в пользу библиотечной реализации.

Тем не менее, важно отметить, что каждая из представленных моделей имеет свой уникальный подход к решению задачи классификации и свои преимущества, вследствие чего они подходят для использования в ряде различных задач анализа данных. Однако в конкретной задаче классификации наилучшие результаты были показаны одной моделью, которая и была выбрана для финальной версии приложения.

Далее было необходимо протестировать работу выбранной модели на пользовательских данных. На рисунках 27-28 представлено заполнение формы и получение ответа с сервера с предсказанной категорией торнадо.

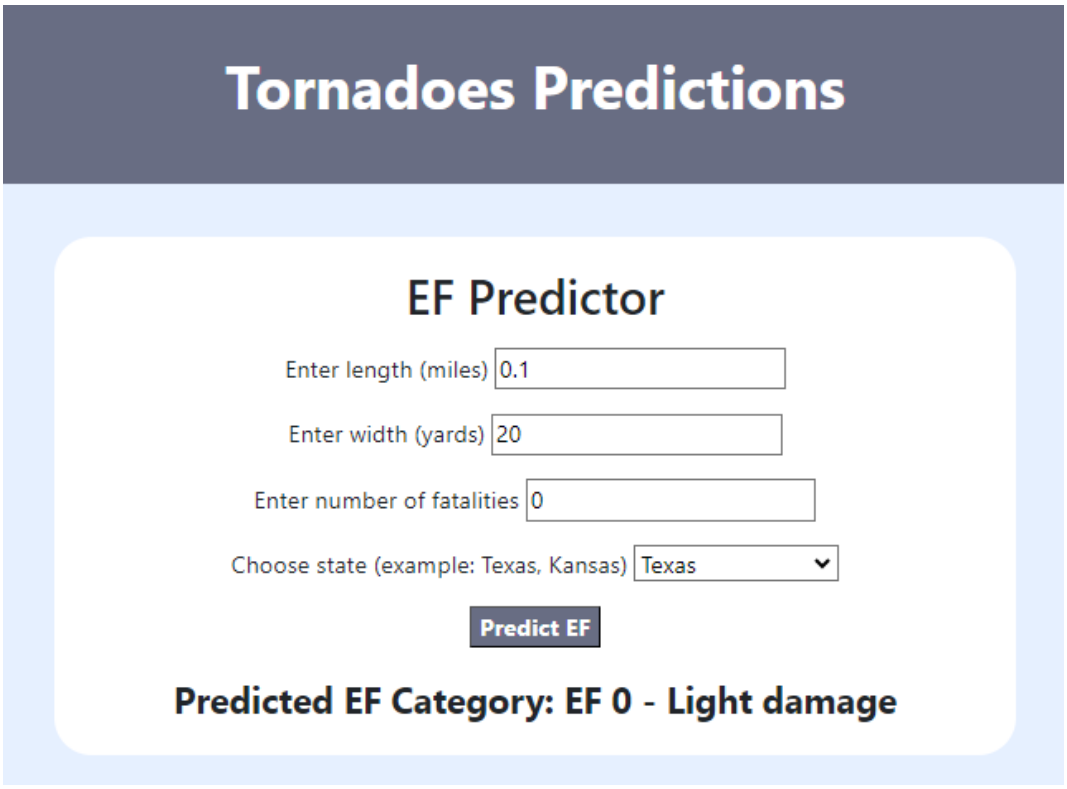

Рисунок 27 - Предсказанная категория торнадо: легкие повреждения

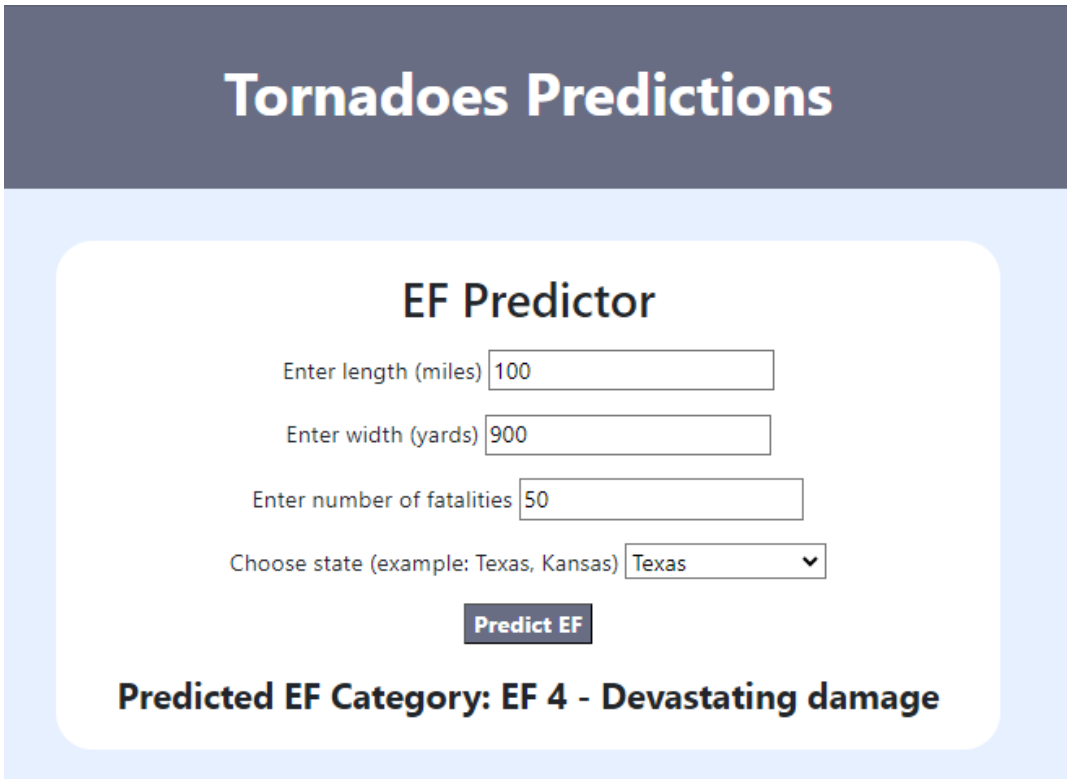

Рисунок 28 - Предсказанная категория торнадо: разрушительный ущерб

Видим, что модель работает корректно. Действительно, с увеличением длины в милях и ширины в ярдах, а также числа погибших, увеличивается индекс категории торнадо по расширенной шкале Фудзиты.

Наконец осталось проверить работоспособность второго объекта пользовательского интерфейса приложения – карты частоты появлений торнадо с в период с 1950 по 2022 год. На рисунках 29-31 продемонстрировано изменение карты в разные годы этого периода.

Карта корректно отображает количество торнадо, зарегистрированных в каждом штате к моменту текущего года. Действительно, самым ярким цветом в 2022 году отмечен штат Техас, который, согласно данным датасета, насчитывает 9265 случаев торнадо с 1950 по 2022 года.

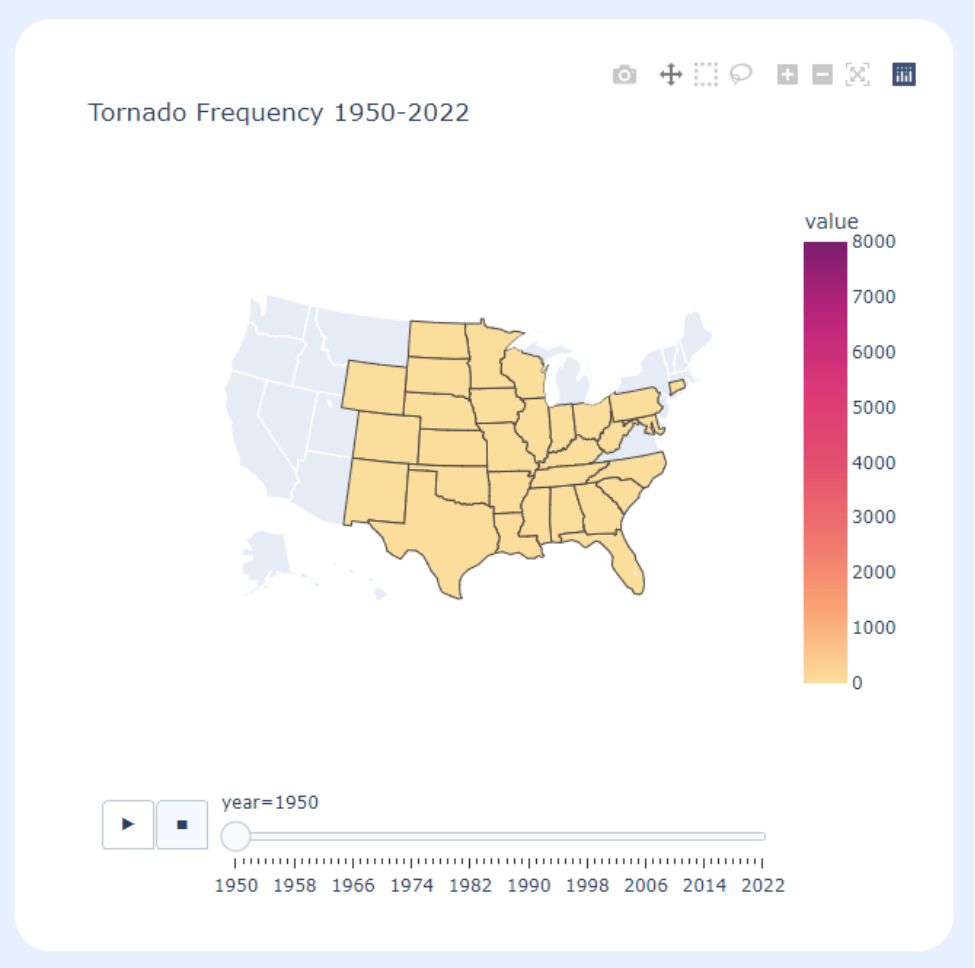

Рисунок 29 - Частота появлений торнадо к 1950 году

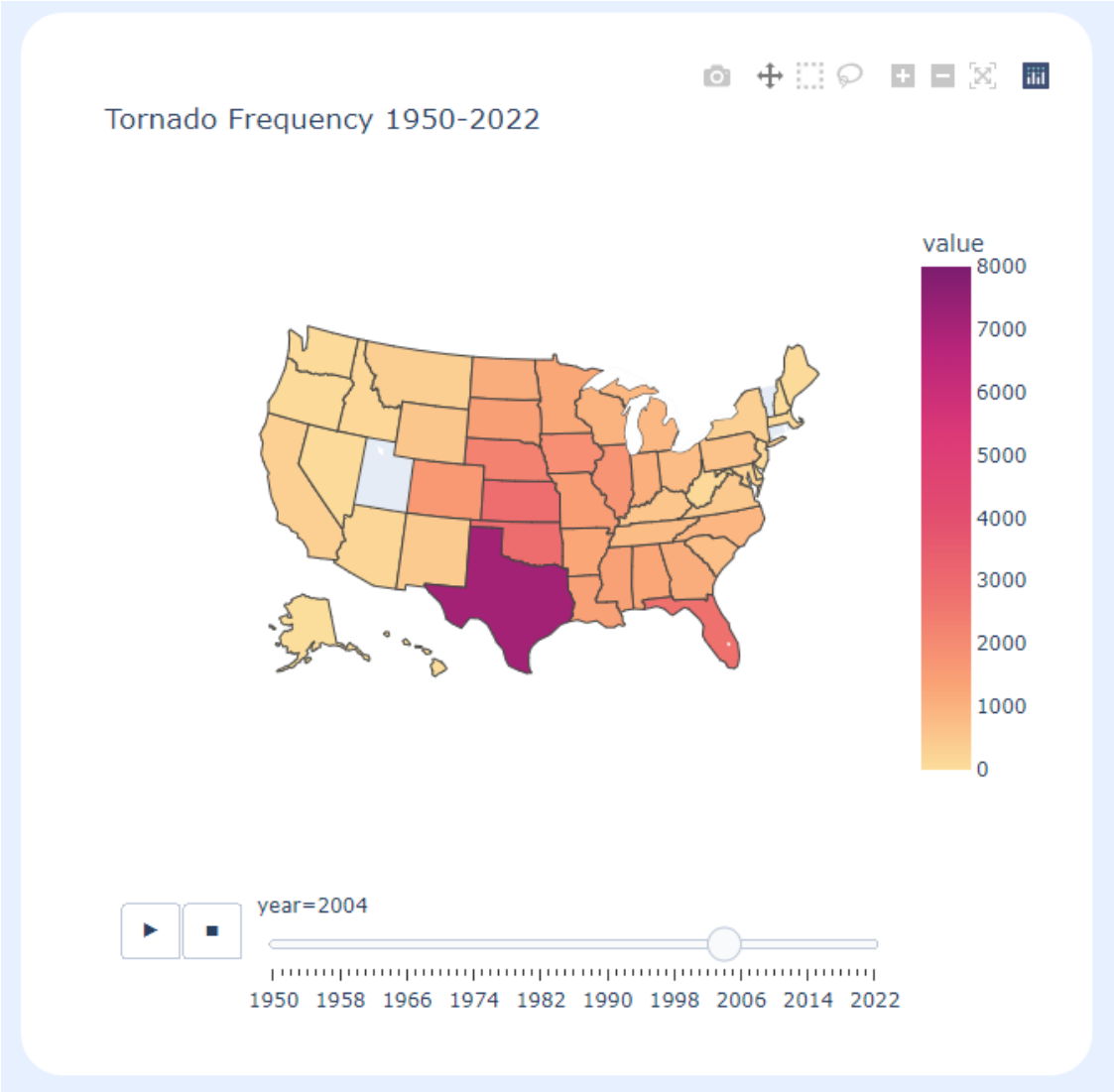

Рисунок 29 - Частота появлений торнадо к 2004 году

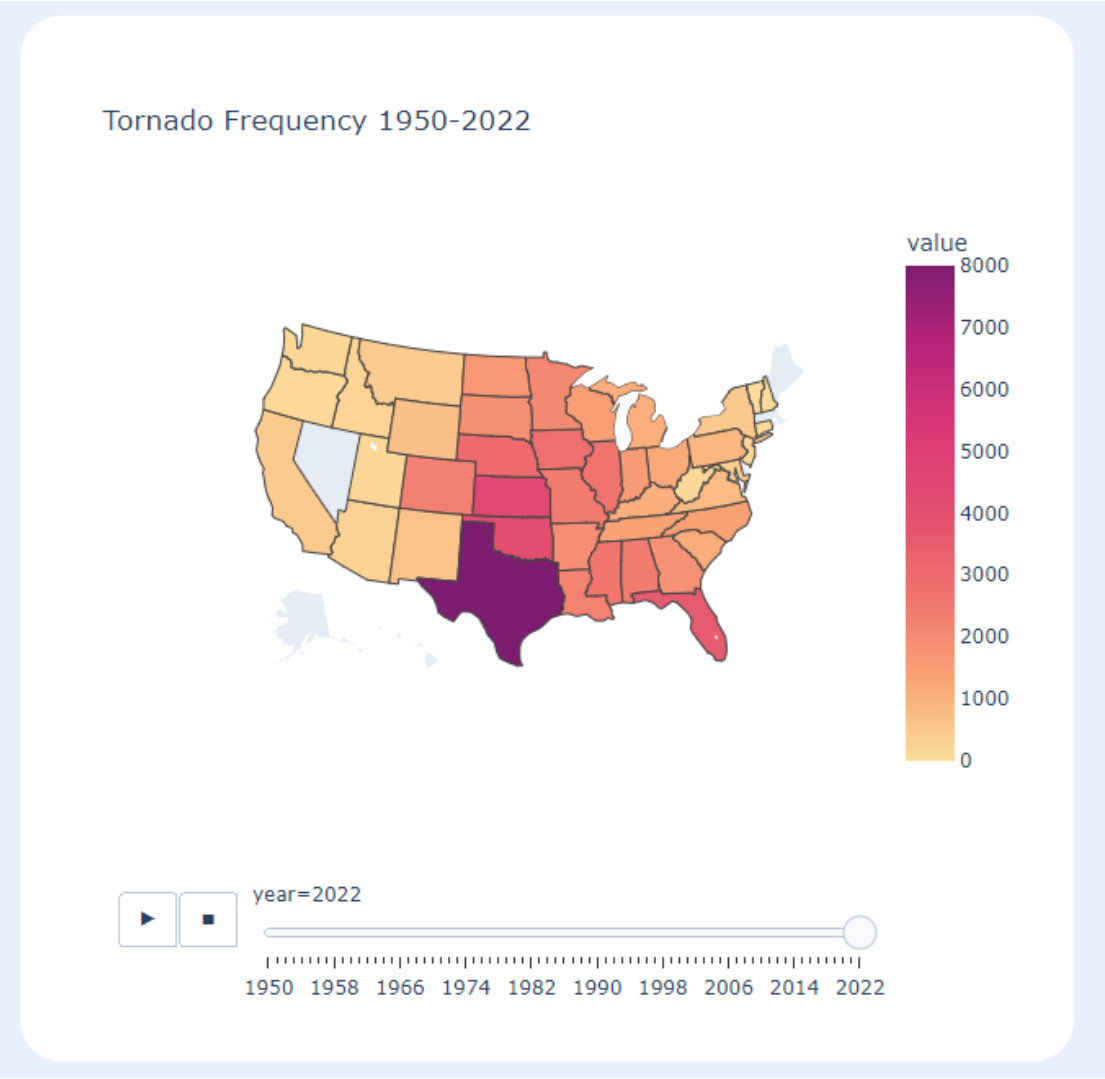

Рисунок 30 - Частота появлений торнадо к 2022 году

ЗАКЛЮЧЕНИЕ

В результате данной курсовой работы было успешно создано приложение для прогнозирования силы торнадо по нескольким характеристикам. Для выполнения прогноза была использована предсказывающая модель – дерево решений с максимальной глубиной 5. По сравнению с другими моделями, чья точность находилась в диапазоне от 0.64 до 0.67, данная модель продемонстрировала самую высокую точность предсказаний на тестовой выборке данных – 0.7.

В процессе работы были изучены ключевые факторы, влияющие на категорию разрушающей силы торнадо. Среди них длина и ширина природного явления, число погибших, начальные широта и долгота места происшествия. Особенно важным оказалось выявление корреляции между числом погибших и целевым признаком модели. Это подтвердило гипотезу о линейной зависимости между разрушающей силой и количеством жертв и позволило убедиться в корректной работе используемой модели.

Нельзя не отметить то, что проект имеет потенциал для дальнейшего развития и совершенствования. Возможными направлениями для будущих исследований являются улучшение точности моделей путем корректировки гиперпараметров модели и использование других методов статистического анализа данных.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. T. Fujita. Tornadoes and Downbursts in the Context of Generalized Planetary Scales. – Journal of the atmospheric sciences, 1981, vol. 38 №8 – 1511-1526 с.
2. The Enhanced Fujita Scale (EF Scale). NOAA`s National Weather Service – URL: www.spc.noaa.gov/efscale/ (дата обращения: 23.12.2023).
3. Учебник по машинному обучению Яндекс – URL: education.yandex.ru/handbook/ml/article/ (дата обращения: 23.12.2032)
4. Gini Impurity – LearnDataSci – URL: www.learndatasci.com/glossary/gini-impurity/ (дата обращения: 23.12.2032)
5. Kaggle dataset Tornados [1950 - 2022] NOAA's National Weather Service Storm Prediction Center – URL: www.kaggle.com/datasets/sujaykapadnis/tornados/data (дата обращения: 23.12.2032)
6. Flask documentation – URL: flask.palletsprojects.com/en/latest/ (дата обращения: 23.12.2032)
7. Pandas documentation – URL: pandas.pydata.org/docs/development/index.html (дата обращения: 23.12.2032)
8. Scikit-learn (Sklearn) documentation – URL: scikit-learn.org/stable/index.html (дата обращения: 24.12.2032)
9. Pickle documentation – URL: docs.python.org/3/library/pickle.html (дата обращения: 24.12.2032)