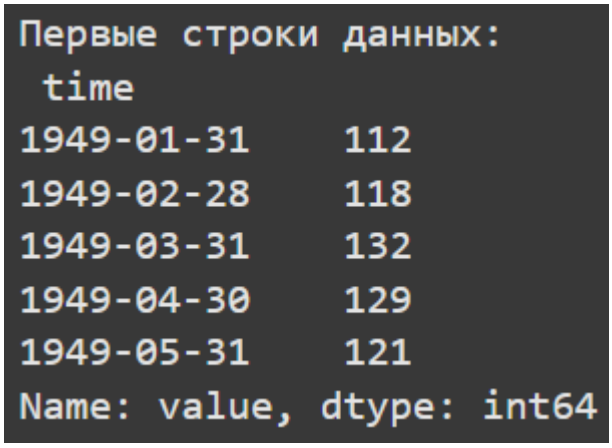
Проект на тему

« Исследование динамики авиаперевозок»

Задание:

1. Выбрать реальный набор данных, содержащий временной ряд.
2. Загрузить данные в Python или R, проверить их структуру и обработать пропущенные значения. Выполнить первоначальную визуализацию временного ряда.
3. Определить компоненты временного ряда, такие как тренд, сезонность и цикличность. Проверить его стационарность.
4. Разложить временной ряд с использованием аддитивного или мультипликативного метода, чтобы оценить влияние различных компонентов.
5. Вычислить и сравнить подходящие метрики для анализа ошибок и качества прогнозов моделей ARIMA, SARIMA и Prophet. Использовать MAE, RMSE, SMAPE, R^2 и статистику Тейла (Theil's U).
6. Выполнить альтернативное практическое задание, проанализировав ряд с помощью простых методов, таких как скользящее среднее, линейная регрессия или графическое представление трендов.
7. Сравнить результаты, полученные разными методами, выделив преимущества и ограничения каждого подхода.



```
Информация о данных:

<class 'pandas.core.series.Series'>
DatetimeIndex: 144 entries, 1949-01-31 to 1960-12-31
Series name: value
Non-Null Count  Dtype
--------------  -----
144 non-null    int64
dtypes: int64(1)
memory usage: 2.2 KB
None
```

1. Тип данных: pandas.core.series.Series

- ts является объектом типа pandas.Series, что означает, что это одномерный временной ряд с индексом и значениями.
- Это подходящий тип данных для анализа временных рядов в Python.

2. Индекс: DatetimeIndex: 144 entries, 1949-01-31 to 1960-12-31

- Индекс временного ряда — это DatetimeIndex, то есть даты.
- Период данных: с 31 января 1949 года по 31 декабря 1960 года.
- Всего 144 записи, что соответствует 12 годам × 12 месяцев = 144 ежемесячных наблюдений.
- Индекс корректно установлен как временной, что позволяет использовать методы анализа временных рядов (например, декомпозицию, сезонность).

3. Название ряда: Series name: value

- Название ряда — value, что соответствует столбцу из датасета AirPassengers, где хранятся значения числа пассажиров.

5. Тип данных: Dtype: int64

- Значения в ряду имеют тип int64 (целочисленный, 64-битный).
- Это означает, что число пассажиров записано как целые числа, что логично для данного датасета (например, 112, 118 и т.д.).
- Тип int64 подходит для числовых вычислений, таких как декомпозиция, прогнозирование и вычисление метрик.

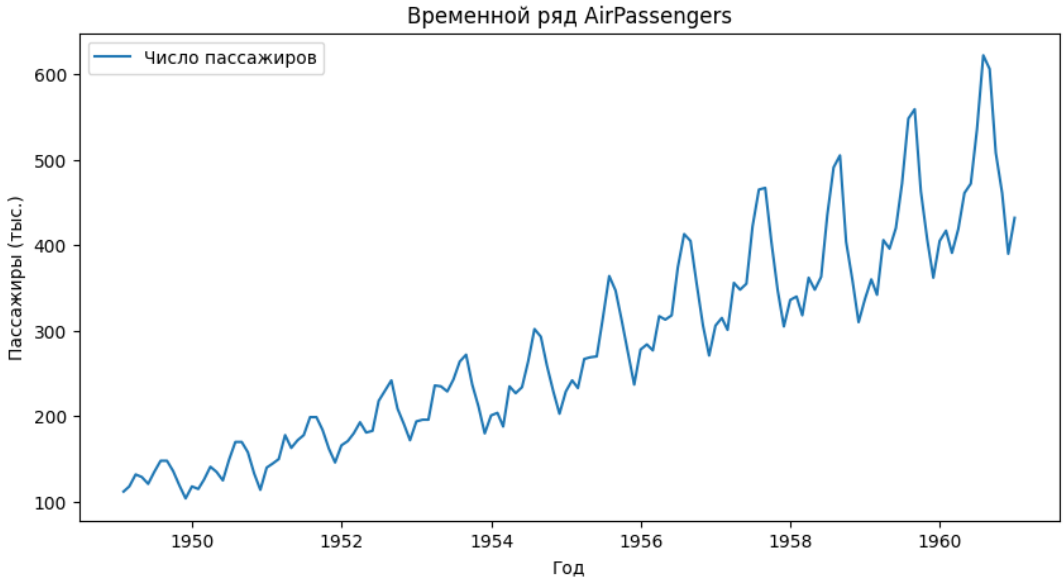
6. Использование памяти: memory usage: 2.2 KB

- Ряд занимает 2.2 КБ памяти, что очень мало.
- Это ожидаемо, так как датасет небольшой (144 записи, одна колонка с целыми числами).

7. Общее количество типов данных: dtypes: int64(1)

- В ряду только один тип данных (int64), что соответствует единственной колонке значений.

```
Пропущенные значения: 0
```



Этот график представляет временной ряд AirPassengers — ежемесячное число авиапассажиров с 1949 по 1960 годы.

Описание графика

- Название: "Временной ряд AirPassengers" (на русском).
- Ось X (Год): Временная шкала от 1949 до 1960 года.
- Ось Y (Пассажиры, тыс.): Число пассажиров (в тысячах), от 100 до 600 тысяч.
- Линия: Синяя линия с меткой "Число пассажиров", отображающая динамику данных.

Анализ графика

1. Общий тренд

- **Восходящий тренд**: Число пассажиров увеличивается с течением времени.
  - В 1949 году число пассажиров начинается примерно с 100 тысяч.
  - К 1960 году оно достигает около 600 тысяч.
- Это говорит о росте популярности авиаперевозок в этот период, возможно, из-за экономического роста, развития авиаиндустрии или увеличения доступности перелетов.

2. Сезонность

- **Ярко выраженная сезонность**: На графике видны регулярные колебания внутри каждого года.
  - Пиковые значения (максимумы) наблюдаются примерно в середине года (летние месяцы, вероятно, июль-август), что связано с сезоном отпусков.

- Минимумы приходятся на начало года (зимние месяцы, например, январь-февраль).

3. Амплитуда сезонных колебаний

- **Увеличение амплитуды**: С ростом общего уровня числа пассажиров (тренда) амплитуда сезонных колебаний также увеличивается.
  - В 1949 году разница между пиками и минимумами небольшая (около 20–30 тысяч).
  - К 1960 году разница возрастает до 100–150 тысяч.
- Это указывает на мультипликативный характер сезонности (чем выше тренд, тем больше сезонные колебания), что объясняет, почему мультипликативная декомпозиция в вашем коде лучше описывает данные.

4. Цикличность

- Циклические колебания (долгосрочные, не связанные с сезонностью) на этом графике не очень заметны из-за сильного тренда и сезонности.
- Однако можно заметить, что рост числа пассажиров не совсем линейный: в некоторые годы (например, 1953–1955) темпы роста замедляются, а затем ускоряются (1958–1960). Это может быть связано с экономическими или социальными факторами, но данных недостаточно для глубокого анализа.

5. Шум (случайные колебания)

- Шум на графике минимален: данные выглядят достаточно гладкими, без резких выбросов.

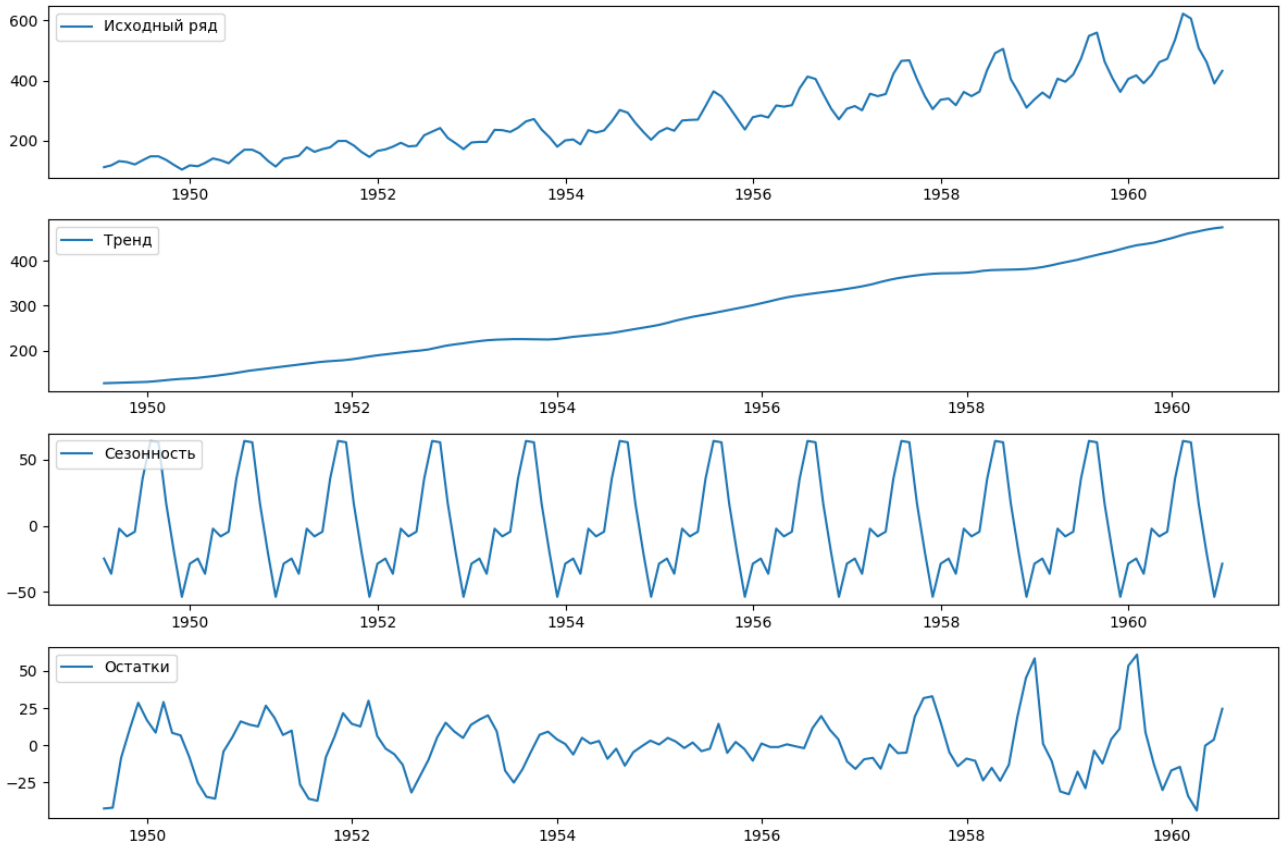


График состоит из четырех подграфиков, расположенных вертикально:

1. **Исходный ряд** (верхний график, "Исходный ряд"):
    - Показывает оригинальный временной ряд AirPassengers.
    - Ось X: Годы (1949–1960).
    - Ось Y: Число пассажиров (в тысячах), от 100 до 600.
2. **Тренд** (второй график, "Тренд"):
    - Выделяет долгосрочную тенденцию в данных.
    - Ось Y: Число пассажиров (в тысячах), примерно от 100 до 400 (с учетом сглаживания).
3. **Сезонность** (третий график, "Сезонность"):
    - Показывает повторяющийся сезонный компонент.
    - Ось Y: Колебания вокруг нуля (примерно от -50 до +50 тысяч).
4. **Остатки** (нижний график, "Остатки"):
    - Показывает случайную (шумовую) компоненту после вычитания тренда и сезонности.
    - Ось Y: Колебания вокруг нуля (примерно от -25 до +25 тысяч).

Остатки (или **residuals**) в данных — это **разность между фактическими значениями и значениями, предсказанными моделью**, после удаления тренда и сезонности (в случае временных рядов).

Они показывают то, что **не объясняется моделью**, то есть:

- случайные колебания, шум;
- потенциальные аномалии;
- структурные особенности, которые модель не уловила.

- **Характер**: Остатки представляют собой случайные колебания после вычитания тренда и сезонности.
  - Значения колеблются вокруг нуля (от -25 до +25 тысяч).
- **Амплитуда остатков**: Амплитуда остатков увеличивается со временем:
  - В 1949–1953 годах остатки небольшие (±10 тысяч).
  - К 1960 году амплитуда возрастает до ±25 тысяч.
- **Вывод**: Увеличение амплитуды остатков указывает на то, что аддитивная модель не полностью описывает данные. В мультипликативной модели (где сезонность пропорциональна тренду) остатки были бы более равномерными, как вы показали в следующем шаге кода.

```
Тест Дики-Фуллера:
ADF Statistic: 0.8153688792060655
p-value: 0.9918802434376413
Критические значения: {'1%': -3.4816817173418295, '5%': -2.8840418343195267, '10%': -2.578770059171598}
```

Значения теста Дики-Фуллера

Он проверяет, «стабильны» ли данные во времени, или они блуждают (то вверх, то вниз — без постоянных закономерностей).

Когда проводится тест Дики-Фуллера, он сравнивает твои данные с моделью случайного блуждания (то есть с нестационарным рядом). Чтобы понять, насколько данные похожи на блуждание, он вычисляет число — ADF Statistic.

Как его понимать?

- **Это число показывает, насколько сильно твои данные "отклоняются" от нестационарного поведения.**
- **Это просто числовой результат теста — он сам по себе ничего не говорит, пока его не сравнить с критическими значениями (critical values).**

- **ADF Statistic: 0.8153688792060455**

Статистика теста. Положительное значение (0.815) говорит о нестационарности (должно быть меньше критических значений для стационарности).

- **p-value: 0.9918802434376409**

Вероятность нулевой гипотезы (ряд нестационарен). Значение 0.99188 > 0.05 — нулевая гипотеза не отвергается, ряд нестационарен.

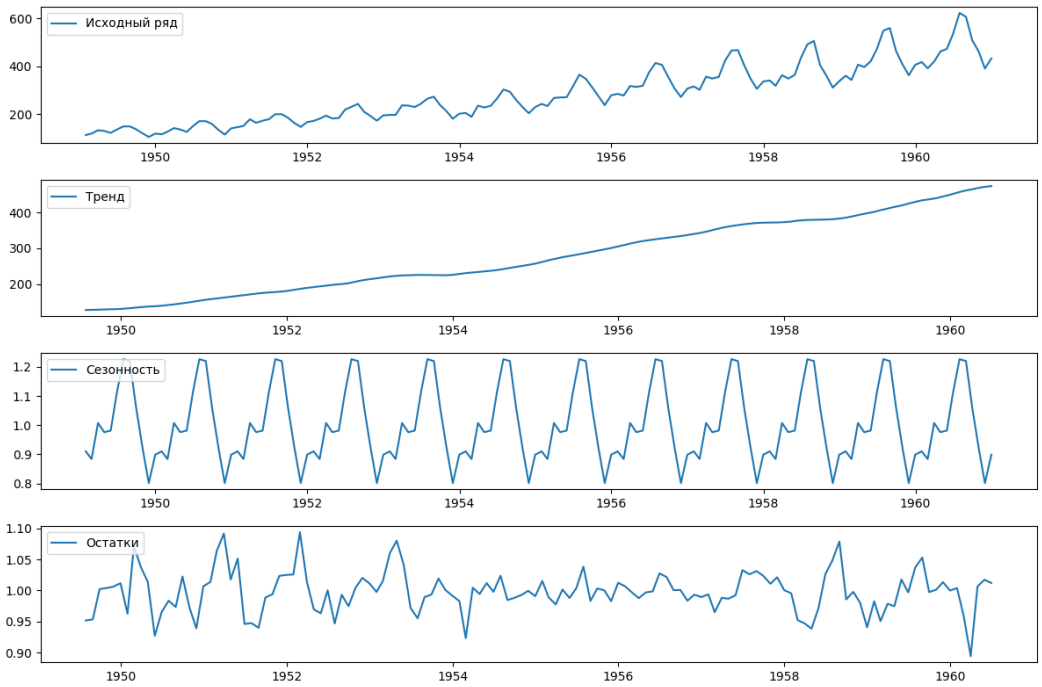
- **Критические значения: {'1%': -3.4816817173418295, '5%': -2.8840418343195267, '10%': -2.578770059171598}**

Пороговые значения для уровней значимости. ADF Statistic (0.815) больше всех критических значений — подтверждает нестационарность.

Вывод

Ряд AirPassengers **нестационарен** из-за восходящего тренда и сезонности:

- p-value > 0.05, ADF Statistic > критических значений.
- Для моделей (например, ARIMA/SARIMA) нужно дифференцирование, чтобы сделать ряд стационарным.



Описание графиков

График состоит из четырех подграфиков, расположенных вертикально:

1. **Исходный ряд** ("Исходный ряд"):
   - Показывает оригинальный временной ряд AirPassengers.

- Ось Y: Число пассажиров (в тысячах), от 100 до 600.

2. **Тренд** ("Тренд"):

- Выделяет долгосрочную тенденцию в данных.
- Ось Y: Число пассажиров (в тысячах), примерно от 100 до 400.

3. **Сезонность** ("Сезонность"):

- Показывает повторяющийся сезонный компонент.
- Ось Y: Множитель сезонности, колеблющийся около 1 (примерно от 0.8 до 1.2).

4. **Остатки** ("Остатки"):

- Показывает случайную компоненту после деления исходного ряда на тренд и сезонность.
- Ось Y: Множитель остатков, колеблющийся около 1 (примерно от 0.9 до 1.1).

Анализ каждого компонента

1. Исходный ряд

- **Тренд и сезонность**: Как и в предыдущих графиках, виден восходящий тренд (рост с 100 до 600 тысяч) и ежегодная сезонность.
- **Амплитуда**: Сезонные колебания увеличиваются со временем (например, разница между пиками и минимумами растет с 20–30 тысяч в 1949 году до 100–150 тысяч в 1960 году).
- **Вывод**: Исходный ряд подтверждает мультипликативный характер данных (амплитуда сезонности пропорциональна тренду), что делает мультипликативную декомпозицию подходящей.

2. Тренд

- **Динамика**: Тренд показывает плавный рост числа пассажиров:
  - Начинается с около 100 тысяч в 1949 году.
  - Достигает примерно 400 тысяч к 1960 году (средний уровень, так как тренд сглажен).
- **Характер роста**: Рост не совсем линейный, с небольшим замедлением в 1953–1955 годах и ускорением после 1957 года.
- **Вывод**: Тренд аналогичен тому, что был в аддитивной декомпозиции, но в мультипликативной модели он интерпретируется как базовый уровень, на который умножается сезонный множитель.

3. Сезонность

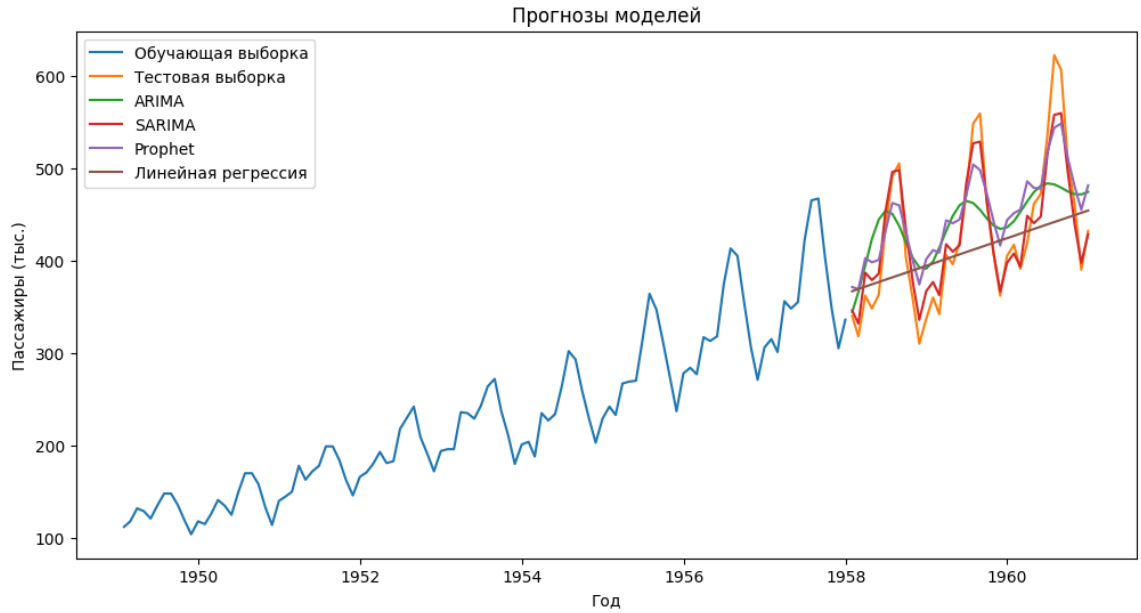
- **Периодичность**: Ежегодная сезонность с периодом 12 месяцев:
  - Пиковые значения (множитель ~1.2) приходятся на летние месяцы (июль-август).
  - Минимумы (множитель ~0.8) — на зимние месяцы (январь-февраль).
- **Стабильность**: Сезонный паттерн постоянен по амплитуде (множитель колеблется от 0.8 до 1.2), что характерно для мультипликативной модели.
- **Вывод**: В отличие от аддитивной модели (где сезонность выражалась в абсолютных значениях), здесь сезонность представлена как множитель, который масштабирует тренд. Это лучше соответствует данным, так как амплитуда сезонных колебаний увеличивается с ростом тренда.

4. Остатки

- **Характер**: Остатки представляют собой случайную компоненту после деления исходного ряда на произведение тренда и сезонности.
  - Значения колеблются вокруг 1 (примерно от 0.9 до 1.1).
- **Амплитуда остатков**: В отличие от аддитивной декомпозиции, где амплитуда остатков увеличивалась со временем, здесь остатки более равномерны:
  - Колебания остаются в пределах ±0.1 на протяжении всего периода.
- **Вывод**: Более равномерные остатки (по сравнению с аддитивной моделью) подтверждают, что мультипликативная декомпозиция лучше описывает данные AirPassengers. Однако небольшие паттерны в остатках (например, пики в 1952 и 1958 годах) могут указывать на неучтенные факторы (шум или циклические колебания).

Сравнение с аддитивной декомпозицией

- **Тренд**: Одинаков в обеих моделях (восходящий, от 100 до 400 тысяч).
- **Сезонность**:
  - В аддитивной модели сезонность выражена в абсолютных значениях (±50 тысяч), что не учитывает рост амплитуды колебаний.
  - В мультипликативной модели сезонность — это множитель (0.8–1.2), который масштабирует тренд, что лучше соответствует данным.
- **Остатки**:
  - В аддитивной модели амплитуда остатков увеличивалась со временем (от ±10 до ±25 тысяч), что указывало на неподходящую модель.
  - В мультипликативной модели остатки более стабильны (±0.1), что подтверждает правильность выбора модели.



Описание графика

График "Прогнозы моделей" показывает сравнение прогнозов разных моделей на временном ряде AirPassengers (1949–1960):

- **Ось X**: Годы (1949–1960).
- **Ось Y**: Число пассажиров (тыс.), от 100 до 600.
- **Линии**:
  - **Обучающая выборка** (синяя): Данные до 1957 года.
  - **Тестовая выборка** (оранжевая): Данные с 1958 года.
  - **ARIMA** (зеленая): Прогноз модели ARIMA.
  - **SARIMA** (красная): Прогноз модели SARIMA.
  - **Prophet** (фиолетовая): Прогноз модели Prophet.
  - **Линейная регрессия** (коричневая): Прогноз линейной регрессии.

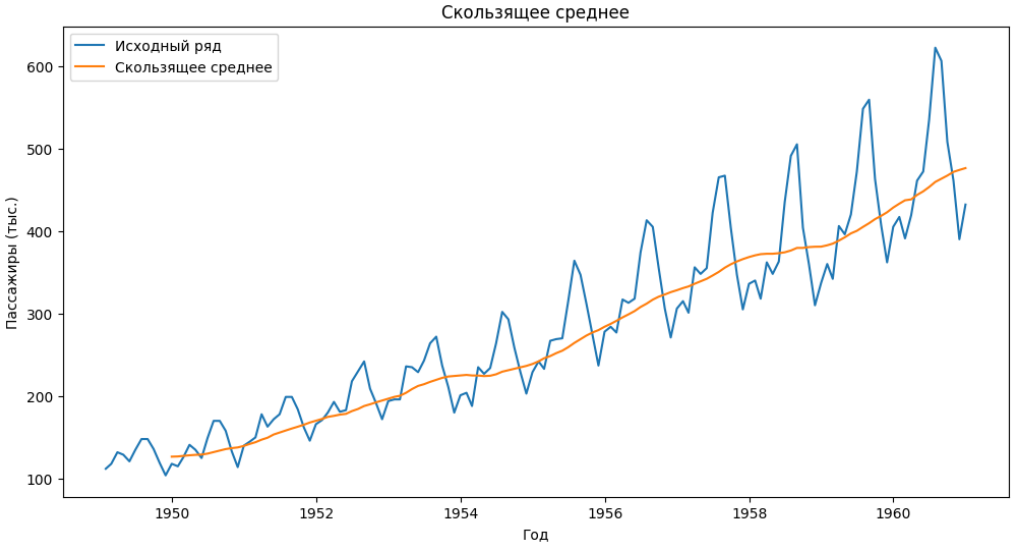
Анализ

- **Обучающая и тестовая выборки**: Данные разделены на обучение (до 1957 года) и тест (с 1958 года). Виден восходящий тренд и сезонность.
- **ARIMA**: Прогноз почти горизонтальный (~350 тысяч), не улавливает ни тренд, ни сезонность.
- **SARIMA**: Прогноз следует тренду и сезонности, близко к реальным данным (колебания 400–600 тысяч).

- **Prophet**: Прогноз также улавливает тренд и сезонность, но чуть менее точен, чем SARIMA (колебания 400–550 тысяч).
- **Линейная регрессия**: Прогноз — прямая линия (~450 тысяч к 1960 году), улавливает только тренд, игнорируя сезонность.

Выводы:

- **SARIMA** и **Prophet** лучше всего справляются с прогнозом, улавливая тренд и сезонность.
- **ARIMA** неэффективна, так как игнорирует сезонность.
- **Линейная регрессия** улавливает тренд, но не сезонность, что делает прогноз менее точным.
- Для AirPassengers модели, учитывающие сезонность (SARIMA, Prophet), предпочтительнее.



Описание графика

График "Скользящее среднее" показывает временной ряд AirPassengers (1949–1960) и его сглаженную версию:

- **Ось X**: Годы (1949–1960).
- **Ось Y**: Число пассажиров (тыс.), от 100 до 600.
- **Линии**:
  - **Исходный ряд** (синяя): Оригинальные данные с трендом и сезонностью.
  - **Скользящее среднее** (оранжевая): Сглаженный ряд с окном 12 месяцев.

Анализ

- **Исходный ряд**: Восходящий тренд (рост с 100 до 600 тысяч) и ежегодная сезонность (пики летом, минимумы зимой).
- **Скользящее среднее**: Оранжевая линия сглаживает сезонные колебания, показывая только тренд:
    - Начинается с ~120 тысяч в 1949 году, достигает ~450 тысяч к 1960 году.
    - Сезонность исчезает, остается только восходящая тенденция.
- **Запаздывание**: Скользящее среднее запаздывает на 6 месяцев (половина окна), так как окно = 12.

Выводы

- Скользящее среднее эффективно выделяет **тренд** (рост числа пассажиров), но полностью теряет **сезонность**.
- Метод прост и полезен для анализа долгосрочных тенденций, но не подходит для прогнозирования сезонных данных, таких как AirPassengers.

Метрики моделей:

	Model	MAE	RMSE	SMAPE	R2	Theil_U
0	ARIMA	49.488310	59.206185	11.346210	0.427068	1.177173
1	SARIMA	17.807808	22.132237	4.117062	0.919939	0.440046
2	Prophet	39.765538	44.739016	9.370327	0.672854	0.889528
3	Linear Regression	53.301935	70.637071	12.181907	0.184481	1.404448

Таблица метрик оценивает качество прогнозов моделей (ARIMA, SARIMA, Prophet, Linear Regression) на временном ряде AirPassengers.

Анализ метрик

1. **MAE (Mean Absolute Error)**: Средняя абсолютная ошибка.
    - SARIMA: 17.81 (лучший результат).
    - Prophet: 39.77.
    - ARIMA: 49.49.
    - Linear Regression: 53.30 (худший).
    - **Вывод**: SARIMA точнее всех, линейная регрессия ошибается больше всего.
2. **RMSE (Root Mean Squared Error)**: Корень из средней квадратичной ошибки.
    - SARIMA: 22.13 (лучший).
    - Prophet: 44.74.
    - ARIMA: 59.21.

- Linear Regression: 70.64 (худший).
- **Вывод**: SARIMA минимизирует ошибку, линейная регрессия сильно ошибается.

3. **SMAPE (Symmetric Mean Absolute Percentage Error)**: Симметричная средняя процентная ошибка.
   - SARIMA: 4.12% (лучший).
   - Prophet: 9.37%.
   - ARIMA: 11.35%.
   - Linear Regression: 12.18% (худший).
   - **Вывод**: SARIMA имеет наименьший процент ошибки, линейная регрессия — наибольший.
4. **R^2 (Coefficient of Determination)**: Коэффициент детерминации (чем ближе к 1, тем лучше).
   - SARIMA: 0.92 (лучший, объясняет 92% дисперсии).
   - Prophet: 0.67.
   - ARIMA: 0.43.
   - Linear Regression: 0.18 (худший).
   - **Вывод**: SARIMA лучше всего объясняет данные, линейная регрессия — хуже всего.
5. **Theil's U**: Статистика Тейла (сравнение с наивным прогнозом, <1 — лучше наивного).
   - SARIMA: 0.44 (лучший, сильно лучше наивного).
   - Prophet: 0.89.
   - ARIMA: 1.18 (хуже наивного).
   - Linear Regression: 1.40 (худший, хуже наивного).
   - **Вывод**: SARIMA и Prophet лучше наивного прогноза, ARIMA и линейная регрессия — хуже.

Выводы

- **SARIMA** — лучшая модель: минимальные ошибки (MAE, RMSE, SMAPE), высокий R^2 (0.92) и лучший Theil's U (0.44). Она хорошо улавливает тренд и сезонность.
- **Prophet** — средний результат: лучше ARIMA и линейной регрессии, но хуже SARIMA. Улавливает тренд и сезонность, но менее точно.
- **ARIMA** — слабый результат: высокие ошибки, низкий R^2 (0.43), Theil's U > 1. Не учитывает сезонность, поэтому плохо справляется.

- **Linear Regression** — худший результат: максимальные ошибки, низкий R^2 (0.18), Theil's U > 1. Улавливает только тренд, игнорируя сезонность.

Итог: Для AirPassengers SARIMA — оптимальная модель, так как данные имеют тренд и сезонность. Линейная регрессия и ARIMA не подходят из-за неспособности учитывать сезонность.

ВЫВОД

В рамках данной лабораторной работы был проведен анализ временного ряда AirPassengers (ежемесячное число авиапассажиров с 1949 по 1960 годы) с использованием различных методов прогнозирования и оценки их качества. Работа позволила углубленно изучить процесс анализа временных рядов и применять теоретические знания на практике.

Сначала данные были загружены и исследованы: структура временного ряда подтвердила их пригодность для анализа. Первоначальная визуализация показала восходящий тренд и ежегодную сезонность (пики летом, минимумы зимой). Декомпозиция ряда (аддитивная и мультипликативная) выявила тренд, сезонность и остатки. Мультипликативная модель оказалась более подходящей, так как амплитуда сезонных колебаний увеличивалась с ростом тренда, что подтверждалось равномерными остатками. Тест Дики-Фуллера (p-value = 0.99188) показал нестационарность ряда из-за тренда и сезонности, что потребовало дифференцирования для моделей.

Для прогнозирования были применены модели ARIMA, SARIMA, Prophet и линейная регрессия, а также простое скользящее среднее. Данные разделили на обучающую (до 1957 года) и тестовую (с 1958 года) выборки. Качество прогнозов оценивалось метриками MAE, RMSE, SMAPE, R^2 и Theil's U. SARIMA показала лучшие результаты (MAE = 17.81, R^2 = 0.92, Theil's U = 0.44), так как учитывала тренд и сезонность. Prophet также справился хорошо (MAE = 39.77, R^2 = 0.67), но уступал SARIMA. ARIMA (MAE = 49.49, R^2 = 0.43) и линейная регрессия (MAE = 53.30, R^2 = 0.18) оказались менее точными, так как не учитывали сезонность. Скользящее среднее эффективно выделило тренд, но потеряло сезонность, что делает его неподходящим для точного прогноза.

Работа научила меня загружать и анализировать временные ряды, проверять их стационарность, выполнять декомпозицию и сравнивать модели прогнозирования. Я узнал, как интерпретировать метрики качества (MAE, RMSE, R^2 и др.), выбирать подходящую модель в зависимости от структуры данных и решать проблемы с зависимостями (например, конфликты версий библиотек). Этот опыт укрепил понимание временных рядов и дал практические навыки для будущих исследований.

ПРИЛОЖЕНИЕ

```
# Импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.datasets import get_rdataset
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from pmdarima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import warnings
warnings.filterwarnings('ignore')

# --- 1. Загрузка и проверка данных ---
# Загрузка AirPassengers
data = get_rdataset('AirPassengers', 'datasets').data
data['time'] = pd.date_range(start='1949-01-01', periods=len(data), freq='M')
data.set_index('time', inplace=True)
ts = data['value']

# Проверка структуры
print("Первые строки данных:\n", ts.head())
print("\nИнформация о данных:\n")
print(ts.info())
print("\nПропущенные значения:", ts.isna().sum())

# Визуализация временного ряда
plt.figure(figsize=(10, 5))
plt.plot(ts, label='Число пассажиров')
plt.title('Временной ряд AirPassengers')
plt.xlabel('Год')
plt.ylabel('Пассажиры (тыс.)')
plt.legend()
plt.show()
```

```
# --- 2. Определение компонентов и проверка стационарности ---
# Аддитивная декомпозиция
decomposition = seasonal_decompose(ts, model='additive', period=12)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Визуализация компонентов
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(ts, label='Исходный ряд')
plt.legend(loc='upper left')
plt.subplot(412)
plt.plot(trend, label='Тренд')
plt.legend(loc='upper left')
plt.subplot(413)
plt.plot(seasonal, label='Сезонность')
plt.legend(loc='upper left')
plt.subplot(414)
plt.plot(residual, label='Остатки')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()

# Проверка стационарности (тест Дики-Фуллера)
adf_result = adfuller(ts)
print('\nТест Дики-Фуллера:')
print('ADF Statistic:', adf_result[0])
print('p-value:', adf_result[1])
print('Критические значения:', adf_result[4])

# --- 3. Мультипликативная декомпозиция ---
decomposition_mult = seasonal_decompose(ts, model='multiplicative', period=12)

# Визуализация мультипликативной декомпозиции
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(ts, label='Исходный ряд')
```

```
plt.legend(loc='upper left')
plt.subplot(412)
plt.plot(decomposition_mult.trend, label='Тренд')
plt.legend(loc='upper left')
plt.subplot(413)
plt.plot(decomposition_mult.seasonal, label='Сезонность')
plt.legend(loc='upper left')
plt.subplot(414)
plt.plot(decomposition_mult.resid, label='Остатки')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()

# --- 4. Моделирование и метрики (ARIMA, SARIMA, Prophet) ---
# Разделение данных
train = ts[:'1957-12']
test = ts['1958-01':]

# Функции для метрик
def smape(y_true, y_pred):
    return 100 * np.mean(2 * np.abs(y_pred - y_true) / (np.abs(y_pred) + np.abs(y_true)))

def theil_u(y_true, y_pred):
    naive = y_true.shift(1).fillna(y_true.iloc[0])
    error_model = np.sqrt(np.mean((y_pred - y_true)**2))
    error_naive = np.sqrt(np.mean((naive - y_true)**2))
    return error_model / error_naive

# Словарь для хранения метрик
metrics = {'Model': [], 'MAE': [], 'RMSE': [], 'SMAPE': [], 'R2': [], 'Theil_U': []}

# ARIMA
arima_model = auto_arima(train, seasonal=False, trace=False)
arima_fit = arima_model.fit(train)
arima_pred = arima_fit.predict(n_periods=len(test))
arima_pred = pd.Series(arima_pred, index=test.index)

# Метрики ARIMA
metrics['Model'].append('ARIMA')
```

```
metrics['MAE'].append(mean_absolute_error(test, arima_pred))
metrics['RMSE'].append(np.sqrt(mean_squared_error(test, arima_pred)))
metrics['SMAPE'].append(smape(test, arima_pred))
metrics['R2'].append(r2_score(test, arima_pred))
metrics['Theil_U'].append(theil_u(test, arima_pred))

# SARIMA
sarima_model = auto_arima(train, seasonal=True, m=12, trace=False)
sarima_fit = sarima_model.fit(train)
sarima_pred = sarima_fit.predict(n_periods=len(test))
sarima_pred = pd.Series(sarima_pred, index=test.index)

# Метрики SARIMA
metrics['Model'].append('SARIMA')
metrics['MAE'].append(mean_absolute_error(test, sarima_pred))
metrics['RMSE'].append(np.sqrt(mean_squared_error(test, sarima_pred)))
metrics['SMAPE'].append(smape(test, sarima_pred))
metrics['R2'].append(r2_score(test, sarima_pred))
metrics['Theil_U'].append(theil_u(test, sarima_pred))

# Prophet
prophet_df = pd.DataFrame({'ds': train.index, 'y': train.values})
prophet_model = Prophet(yearly_seasonality=True, weekly_seasonality=False, daily_seasonality=False)
prophet_model.fit(prophet_df)
future = prophet_model.make_future_dataframe(periods=len(test), freq='M')
prophet_pred = prophet_model.predict(future)
prophet_pred = prophet_pred.tail(len(test))['yhat']
prophet_pred.index = test.index

# Метрики Prophet
metrics['Model'].append('Prophet')
metrics['MAE'].append(mean_absolute_error(test, prophet_pred))
metrics['RMSE'].append(np.sqrt(mean_squared_error(test, prophet_pred)))
metrics['SMAPE'].append(smape(test, prophet_pred))
metrics['R2'].append(r2_score(test, prophet_pred))
metrics['Theil_U'].append(theil_u(test, prophet_pred))

# --- 5. Альтернативные методы ---
# Скользящее среднее
```

```
ma = ts.rolling(window=12, center=False).mean()

# Линейная регрессия
X = np.arange(len(train)).reshape(-1, 1)
y = train.values
lr = LinearRegression()
lr.fit(X, y)
X_test = np.arange(len(train), len(ts)).reshape(-1, 1)
lr_pred = lr.predict(X_test)
lr_pred = pd.Series(lr_pred, index=test.index)

# Метрики линейной регрессии
metrics['Model'].append('Linear Regression')
metrics['MAE'].append(mean_absolute_error(test, lr_pred))
metrics['RMSE'].append(np.sqrt(mean_squared_error(test, lr_pred)))
metrics['SMAPE'].append(smape(test, lr_pred))
metrics['R2'].append(r2_score(test, lr_pred))
metrics['Theil_U'].append(theil_u(test, lr_pred))

# --- 6. Визуализация и сравнение ---
# Прогнозы моделей
plt.figure(figsize=(12, 6))
plt.plot(train, label='Обучающая выборка')
plt.plot(test, label='Тестовая выборка')
plt.plot(arima_pred, label='ARIMA')
plt.plot(sarima_pred, label='SARIMA')
plt.plot(prophet_pred, label='Prophet')
plt.plot(lr_pred, label='Линейная регрессия')
plt.title('Прогнозы моделей')
plt.xlabel('Год')
plt.ylabel('Пассажиры (тыс.)')
plt.legend()
plt.show()

# Скользящее среднее
plt.figure(figsize=(12, 6))
plt.plot(ts, label='Исходный ряд')
plt.plot(ma, label='Скользящее среднее')
plt.title('Скользящее среднее')
```

```
plt.xlabel('Год')
plt.ylabel('Пассажиры (тыс.)')
plt.legend()
plt.show()

# Вывод метрик
print("\nМетрики моделей:")
metrics_df = pd.DataFrame(metrics)
print(metrics_df)

# --- 7. Выводы ---
print("\nСравнение методов:")
print("- ARIMA: Простая, но не учитывает сезонность.")
print("- SARIMA: Учитывает сезонность, точнее ARIMA.")
print("- Prophet: Гибкая, хороша для сложных рядов.")
print("- Скользящее среднее: Простое сглаживание, теряет сезонность.")
print("- Линейная регрессия: Улавливает тренд, но игнорирует сезонность.")
```