МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра информационных систем и технологий

ОТЧЕТ ПО ПРАКТИКЕ

Вид практики учебная
(учебная, производственная)

Тип практики практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков в научно-исследовательской деятельности
(в соответствии с ОПОП ВО)

Сроки прохождения практики: с 01.07.2019 по 13.07.2019
(в соответствии с календарным учебным графиком)

по направлению подготовки 09.03.01 Информатика и вычислительная техника
(уровень бакалавриата)
направленность (профиль) «Информационные системы»

Студент группы № 6203-090301D_____ Г.В.Власов
Руководитель практики,
доцент кафедры ИСТ, к.т.н._____ А.А. Столбова

Дата сдачи 13.07.2019
Дата защиты 13.07.2019

Оценка _____

Самара 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра информационных систем и технологий

Индивидуальное задание на практику

Студенту Власову Георгию Владимировичу группы № 6203-090301D.
Направление на практику оформлено приказом по университету от __.06.2019 г. № ___-ст
на_____кафедру информационных систем и технологий_____.
(наименование профильной организации или структурного подразделения университета)

Планируемые результаты освоения образовательной программы (компетенции)	Планируемые результаты практики	Содержание задания
ПК-1 способностью разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина"	Знать: теоретические и методические основы моделирования компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина". Уметь: учитывать влияние технологии программной реализации на процесс разработки моделей компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина". Владеть: навыками разработки моделей информационно-	Проанализировать имеющиеся аналоги. Разработать программную модель компонента информационной системы для анализа метеоусловий.

	логических проектов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина", а также навыками разработки соответствующих методических документов.	
ПК-3 способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	Знать: методологию подготовки заданий и разработки информационно-логических проектов информационных систем. Уметь: обосновывать разрабатываемые информационно-логические проекты информационных систем и разрабатывать соответствующие методические документы. Владеть: навыками разработки информационно-логических проектов информационных систем и соответствующих методических документов.	Обосновать выбор методологии и технологии разработки информационно-логических проектов информационных систем. Разработать информационно-логический проект системы для анализа метеоусловий.

Дата выдачи задания 01.07.2019.
Срок представления на кафедру отчета о практике 13.07.2019.

Руководитель практики,
доцент кафедры ИСТ, к.т.н. _____ А.А. Столбова
(подпись)

Задание принял к исполнению
студент группы № 6203-090301D_____ Г.В.Власов
(подпись)

Рабочий график (план) проведения практики

Дата (период)	Содержание задания	Результаты практики
01.07.2019	Оформить индивидуальное задание на практику.	Оформлено и утверждено индивидуальное задание на практику и рабочий график.
03.07.2019	Проанализировать имеющиеся аналоги.	Обучающийся изучил существующие системы для анализа метеоусловий.
04.07.2019	Разработать программную модель компонента информационной системы для анализа метеоусловий.	Обучающийся приобрёл знания основ программного моделирования, навыки разработки алгоритмов.
06.07.2019	Обосновать выбор методологии и технологии разработки информационно-логических проектов информационных систем.	Обучающийся освоил умение обосновывать выбор методологии и технологии разработки информационно-логических проектов информационных систем.
9.07.2019	Разработать информационно-логический проект системы для анализа метеоусловий.	Обучающийся получил навыки разработки информационно-логического проекта системы для анализа метеоусловий.
11.07.2019	Оформить письменный отчёт по практике.	Обучающийся применил полученные навыки для разработки письменного отчёта по практике. Обучающимся был оформлен и предъявлен руководителю письменный отчет по практике.
12.07.2019	Подготовить устный отчет о прохождении практики.	Подготовлен устный отчет о прохождении практики.
13.07.2019	Защитить отчет по практике	Получен зачёт по практике.

Руководитель практики,
доцент кафедры ИСТ, к.т.н. _____ А.А. Столбова
(подпись)

1.Анализ имеющихся аналогов

1.1.Обзор алгоритмов и методов решения поставленной задачи

Машинное обучение (англ. machine learning, ML) - класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Алгоритмы машинного обучения можно описать как обучение целевой функции f, которая наилучшим образом соотносит входные переменные X и выходную переменную

$$Y: Y = f(X)$$

Мы не знаем, что из себя представляет функция f. Ведь если бы знали, то использовали бы ее напрямую, а не пытались обучить с помощью различных алгоритмов.

Наиболее распространенной задачей в машинном обучении является предсказание значений Y для новых значений X. Это называется прогностическим моделированием, и наша цель - сделать как можно более точное предсказание.

Перечислим некоторые из существующих алгоритмов в машинном обучении.

Линейная регрессия - пожалуй, один из наиболее известных и понятных алгоритмов в статистике и машинном обучении.

Прогностическое моделирование в первую очередь касается минимизации ошибки модели или, другими словами, как можно более точного прогнозирования. Мы будем заимствовать алгоритмы из разных областей, включая статистику, и использовать их в этих целях.

Линейную регрессию можно представить в виде уравнения, которое описывает прямую, наиболее точно показывающую взаимосвязь между входными переменными X и выходными переменными Y. Для составления этого уравнения нужно найти определенные коэффициенты B для входных переменных

Авторегрессионная (AR-) модель (англ. autoregressive model) - модель временных рядов, в которой значения временного ряда в данный момент линейно зависят от

предыдущих значений этого же ряда. Авторегрессионный процесс порядка p (AR(p)-процесс) определяется следующим образом:

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t,$$

где a1,…,ai - параметры модели (коэффициенты авторегрессии);

c- постоянная (часто для упрощения предполагается равной нулю);Et - белый шум.

Модель скользящего среднего q-го порядкаMA(q) - модель временного ряда вида:

$$X_t = \sum_{j=0}^{q} b_j \varepsilon_{t-j},$$

где Et - белый шум;b_j - параметры модели.

Также в модель иногда добавляют константу. Тем не менее, поскольку чаще всего модели скользящего среднего используются для моделирования случайных ошибок временных рядов, то константу можно считать параметром основной модели.

Процесс белого шума формально можно считать процессом скользящего среднего нулевого порядка - MA(0).

Чаще всего на практике используют процесс скользящего среднего первого порядка MA(1).

Модель авторегрессии - скользящего среднего (англ. autoregressive moving-average model, ARMA) - одна из математических моделей, использующихся для анализа и прогнозирования стационарных временных рядов в статистике. Модель ARMA обобщает две более простые модели временных рядов - модель авторегрессии (AR) и модель скользящего среднего (MA).

Моделью ARMA(p, q), где p и q - целые числа, задающие порядок модели, называется следующий процесс генерации временного ряда {Xt}:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i},$$

где c - константа;Et - белый шум, то есть последовательность независимых и одинаково распределенных случайных величин (как правило, нормальных), с нулевым средним;

α, β - действительные числа, авторегрессионные коэффициенты и коэффициенты скользящего среднего, соответственно.

Такая модель может интерпретироваться как линейная модель множественной регрессии, в которой в качестве объясняющих переменных выступают прошлые значения самой зависимой переменной, а в качестве регрессионного остатка - скользящие средние из элементов белого шума. ARMA-процессы имеют более сложную структуру по сравнению со схожими по поведению AR или MA-процессами в чистом виде, но при этом ARMA - процессы характеризующиеся меньшим количеством параметров, что является одним из их преимуществ.

ARIMA (англ. autoregressive integrated moving average, иногда модель Бокса - Дженкинса, методология Бокса - Дженкинса) - интегрированная модель авторегрессии - скользящего среднего - модель и методология анализа временных рядов. Является расширением моделей ARMA для нестационарных временных рядов, которые можно сделать стационарными взятием разностей некоторого порядка от исходного временного ряда (так называемые интегрированные или разностно-стационарные временные ряды). Модель ARIMA(p,d,q) означает, что разности временного ряда порядка d подчиняются модели ARMA(p,q):

$$\triangle^d X_t = c + \sum_{i=1}^{p} a_i \triangle^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t \,,$$

1.2. Описание функций моделируемого компонента информационной системы

Информационная система - взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Моделируемый компонент информационной системы должен осуществить прогнозирование характеристик транспортных поток, таких как: скорость передвижения, время передвижения и количество машин.

Исходные данные представлены в виде таблицы:

trafficData158505

status	avgMeasuredTime	avgSpeed	extID	medianMeasuredTime	TIMESTAMP	vehicleCount	_id	REPORT_ID
OK	141	52	616	141	2014-08-01T07:50:00	1	20746168	158505
OK	141	52	616	141	2014-08-01T07:55:00	0	20746340	158505
OK	119	62	616	119	2014-08-01T08:00:00	1	20746671	158505
OK	119	62	616	119	2014-08-01T08:05:00	1	20747120	158505
OK	119	62	616	119	2014-08-01T08:10:00	1	20747493	158505
OK	119	62	616	119	2014-08-01T08:15:00	3	20747942	158505
OK	99	75	616	99	2014-08-01T08:20:00	3	20748391	158505
OK	126	59	616	126	2014-08-01T08:25:00	3	20748840	158505
OK	126	59	616	126	2014-08-01T08:30:00	3	20749289	158505
OK	135	55	616	135	2014-08-01T08:35:00	2	20749738	158505
OK	135	55	616	135	2014-08-01T08:40:00	3	20750187	158505
OK	135	55	616	135	2014-08-01T08:45:00	1	20750636	158505
OK	83	89	616	83	2014-08-01T08:50:00	2	20751085	158505

Конечный результат должен быть представлен в виде графика, на котором изображены реальные и предсказываемые данные.

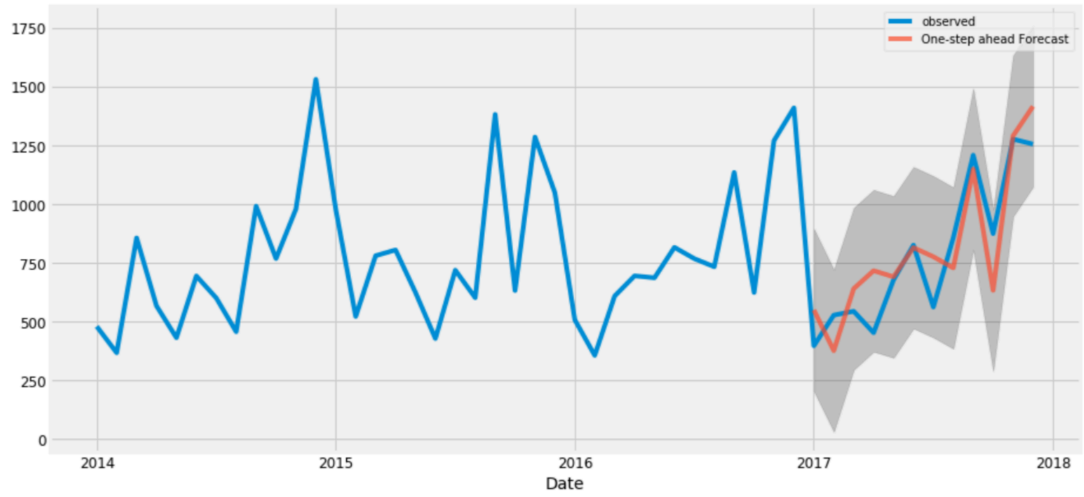


Рисунок 1 – Реальные и предсказываемые данные

1.3. Выбор средств программной реализации

Python - это самый популярный высокоуровневый язык программирования с динамической семантикой. Он довольно прост для работы и чтения: его использование снижает стоимость разработки и обслуживания программ.

Помимо простоты, у Python есть еще один плюс - он довольно легко взаимодействует с другими языками, особенно с C и C++.

По сути, машинное обучение - это технология, которая помогает приложениям на основе искусственного интеллекта обучаться и выдавать результаты автоматически, без человеческого вмешательства.

Python лучше всего подходит для выполнения таких задач, потому что он довольно понятный по сравнению с другими языками. Более того, у него отличная производительность при обработке данных.

Одна из основных причин, почему Python используется для машинного обучения состоит в том, что у него есть множество фреймворков, которые упрощают процесс написания кода и сокращают время на разработку.

В научных расчетах используется Numpy, в продвинутых вычислениях - SciPy, в извлечении и анализе данных - SciKit-Learn. Эти библиотеки работают в таких фреймворках, как TensorFlow, CNTK и Apache Spark.

Существует фреймворк для Python, разработанный специально для машинного обучения - это PyTorch.

Python - самый высокоуровневый и понятный язык, с которым удобно работать. Благодаря его лаконичности и удобству чтения он хорошо подходит для обучения разработке ПО.

Кроме того, Python хорошо подходит для машинного обучения, потому что сами алгоритмы машинного обучения сложны для понимания. При работе с Python разработчику не нужно уделять много внимания непосредственно написанию кода: все внимание он может сосредоточить на решении более сложных задач, связанных с машинным обучением.

Простой синтаксис языка Python помогает разработчику тестировать сложные алгоритмы с минимальной тратой времени на их реализацию.

Еще одно преимущество Python - это обширная поддержка и качественная документация. Существует множество полезных ресурсов о Python, на которых программист может получить помощь и консультацию, находясь на любом этапе разработки.

Следующее преимущество Python в машинном обучении состоит в его гибкости: например, у разработчика есть выбор между объектно-ориентированным подходом и скриптами. Python помогает объединять различные типы данных. Более того, Python особенно удобен для тех разработчиков, которые большую часть кода пишут с помощью IDE.

Средой разработки был выбран Jupyter Notebook – это крайне удобный инструмент для создания красивых аналитических отчетов, так как он позволяет хранить вместе код, изображения, комментарии, формулы и графики.

Jupyter может интерактивно управлять параллельными кластерами, используя асинхронные статусы обратных вызовов и/или MPI. IPython может использоваться как замена стандартной командной оболочки операционной системы, особенно на платформе Windows, возможности оболочки которой ограничены. Поведение по умолчанию похоже на поведение оболочек UNIX-подобных систем, но тот факт, что работа происходит в окружении Python, позволяет добиваться большей настраиваемости и гибкости.

Начиная с версии 4.0, монолитный код был разбит на модули, и независимые от языка модули были выделены в отдельный проект Jupyter.

2. Разработка модели компонента информационной системы, реализующей основные функции компонента системы

2.1 Математическая модель компонента информационной системы

Для выполнения поставленной задачи было решено использовать алгоритмы ARIMA и SARIMA.

ARIMA(p, d, q) является расширением модели ARMA(p, q) для нестационарных временных рядов, которые сводятся к стационарным взятием разностей d-го порядка. Модель имеет следующий вид:

$$\Delta^d Y_t = \sum_{i=1}^{p} \alpha_i \Delta^d Y_{t-i} + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} + \varepsilon_t,$$

где Δd – оператор разности d-го порядка (ΔYt = Yt–Yt−1 – разности 1-го порядка).

Модель SARIMA(p, d, q)(P, D, Q)s - обобщение ARIMA-модели на временные ряды, в которых имеется ярко выраженная сезонная компонента. Дополнительно в такой модели вводятся сезонные параметры (P, D, Q, s), позволяющие учесть циклические колебания процесса.

Описав SARIMA-модель данного временного ряда, можно строить прогноз его будущих значений. Предполагается, что при более детальном изучении, описанный подход к решению поставленной задачи даст неплохие результаты на исходных данных.

2.2 Программная модель компонента информационной системы

```
import numpy as np
import pandas as pd
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.ar_model import AR
from sklearn.metrics import mean_squared_error
from pandas import datetime
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt
import itertools
import warnings
import pmdarima as pm
from scipy import stats
import statsmodels.api as sm
from pmdarima.arima.utils import ndiffs
```

Рисунок 2 - Библиотеки, используемые в программе

```
def parser(x):
    return datetime.strptime(x,'%Y-%m-%dT%X')
```

```
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    me = np.mean(forecast - actual)             # ME
    mae = np.mean(np.abs(forecast - actual))    # MAE
    mpe = np.mean((forecast - actual)/actual)   # MPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr
    mins = np.amin(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    maxs = np.amax(np.hstack([forecast[:,None],
                              actual[:,None]]), axis=1)
    minmax = 1 - np.mean(mins/maxs)             # minmax
    acf1 = acf(fc-test1)[1]                      # ACF1
    return({'mape':mape, 'me':me, 'mae': mae,
            'mpe': mpe, 'rmse':rmse, 'acf1':acf1,
            'corr':corr, 'minmax':minmax})
```

Рисунок 3 - Функции, используемые в программе

3. Разработка контрольного примера

3.1.Описание исходных данных

```
traffic = pd.read_csv('trafficData158505.csv', parse_dates=[5], date_parser=parser)
```

```
traffic = traffic.drop(columns='status')
traffic = traffic.drop(columns='extID')
traffic = traffic.drop(columns='medianMeasuredTime')
traffic = traffic.drop(columns='_id')
traffic = traffic.drop(columns='REPORT_ID')
```

```
traffic = traffic.rename(columns={'TIMESTAMP': 'Дата','avgMeasuredTime': 'Среднее вр
```

```
new_traffic = traffic.iloc[:-16854]
```

```
new_traffic.head()
```

	Среднее время	Средняя скорость	Дата	Кол-во машин
0	141	52	2014-08-01 07:50:00	1
1	141	52	2014-08-01 07:55:00	0
2	119	62	2014-08-01 08:00:00	1
3	119	62	2014-08-01 08:05:00	1
4	119	62	2014-08-01 08:10:00	1

```
new_traffic.tail()
```

	Среднее время	Средняя скорость	Дата	Кол-во машин
95	133	55	2014-08-01 15:45:00	1
96	104	71	2014-08-01 15:50:00	1
97	104	71	2014-08-01 15:55:00	1
98	104	71	2014-08-01 16:00:00	0
99	109	68	2014-08-01 16:05:00	1

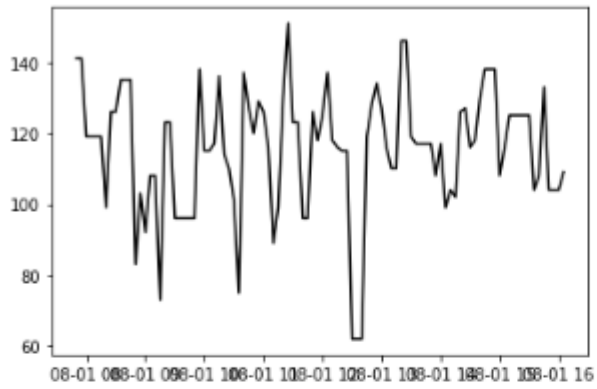
```
new_traffic.shape
```

```
(100, 4)
```

Рисунок 4 – Исходные данные

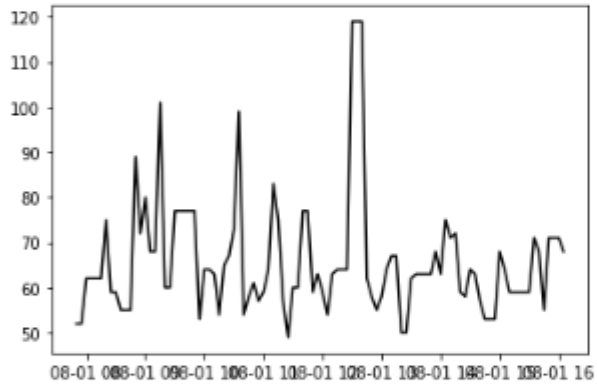
```
plt.plot(new_traffic['Дата'], new_traffic['Среднее время'], color='black')
```

```
[<matplotlib.lines.Line2D at 0x121a3f0f0>]
```




```
plt.plot(new_traffic['Дата'], new_traffic['Средняя скорость'], color = 'black')
```

```
[<matplotlib.lines.Line2D at 0x122e9d390>]
```




```
plt.plot(new_traffic['Дата'], new_traffic['Кол-во машин'], color = 'black')
```

```
[<matplotlib.lines.Line2D at 0x122f0bb70>]
```

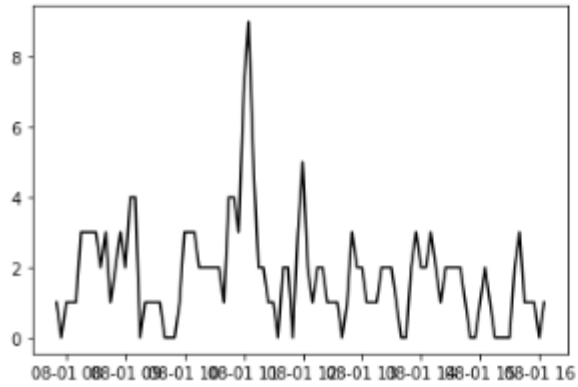


Рисунок 5 – Исходные данные

3.2 Разработка и демонстрация контрольных примеров

Предсказания с использованием алгоритма Arima

Прогнозирование среднего времени:

Среднее время

```
X = new_traffic['Среднее время']

# Original Series
fig, axes = plt.subplots(3, 2)
axes[0, 0].plot(X); axes[0, 0].set_title('Изначальные данные')
plot_acf(X, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(X.diff()); axes[1, 0].set_title('Дифференциирование 1-ого порядка')
plot_acf(X.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(X.diff().diff()); axes[2, 0].set_title('Дифференциирование 2-ого порядка')
plot_acf(X.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```

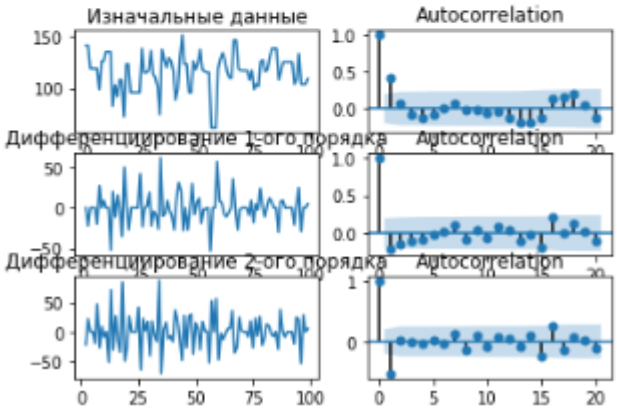


Рисунок 6 – Задание начальных данных и данных для дифференцирования

Использование реализации в пакете ARIMA Statsmodels.

Резюме модели раскрывает много информации. Таблица посередине - это таблица коэффициентов, где значения под 'coef' являются весами соответствующих терминов.

```
model = ARIMA(X, order=(13,1,3))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:        D.Среднее время   No. Observations:                   99
Model:                ARIMA(13, 1, 3)   Log Likelihood                -410.385
Method:                       css-mle   S.D. of innovations             14.433
Date:                Fri, 12 Jul 2019   AIC                            856.770
Time:                        17:56:40   BIC                            903.482
Sample:                             1   HQIC                           875.669

==============================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                          0.0424      0.074      0.577      0.566      -0.102       0.187
ar.L1.D.Среднее время         -1.1465      0.101    -11.298      0.000      -1.345      -0.948
ar.L2.D.Среднее время         -0.1576      0.159     -0.989      0.326      -0.470       0.155
ar.L3.D.Среднее время          0.1902      0.162      1.171      0.245      -0.128       0.508
ar.L4.D.Среднее время         -0.2230      0.162     -1.376      0.173      -0.541       0.095
ar.L5.D.Среднее время         -0.1714      0.164     -1.047      0.298      -0.492       0.150
ar.L6.D.Среднее время         -0.0420      0.166     -0.253      0.801      -0.367       0.283
ar.L7.D.Среднее время          0.0821      0.167      0.492      0.624      -0.245       0.409
ar.L8.D.Среднее время          0.0082      0.166      0.049      0.961      -0.317       0.333
ar.L9.D.Среднее время         -0.0853      0.163     -0.522      0.603      -0.406       0.235
ar.L10.D.Среднее время        -0.0648      0.161     -0.403      0.688      -0.380       0.251
ar.L11.D.Среднее время        -0.0160      0.162     -0.098      0.922      -0.334       0.302
ar.L12.D.Среднее время        -0.0622      0.166     -0.375      0.709      -0.388       0.263
ar.L13.D.Среднее время        -0.1528      0.112     -1.363      0.176      -0.373       0.067
ma.L1.D.Среднее время          0.7713      0.065     11.933      0.000       0.645       0.898
ma.L2.D.Среднее время         -0.7715      0.058    -13.384      0.000      -0.885      -0.659
ma.L3.D.Среднее время         -0.9998      0.069    -14.591      0.000      -1.134      -0.865
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            -0.9142           -0.4193j            1.0058           -0.4316
AR.2            -0.9142           +0.4193j            1.0058            0.4316
AR.3            -1.1103           -0.0000j            1.1103           -0.5000
AR.4            -0.8007           -0.7974j            1.1300           -0.3753
AR.5            -0.8007           +0.7974j            1.1300            0.3753
AR.6            -0.3210           -1.1383j            1.1827           -0.2937
AR.7            -0.3210           +1.1383j            1.1827            0.2937
AR.8             1.1474           -0.3479j            1.1990           -0.0468
AR.9             1.1474           +0.3479j            1.1990            0.0468
AR.10            0.8278           -0.8619j            1.1950           -0.1282
AR.11            0.8278           +0.8619j            1.1950            0.1282
AR.12            0.4124           -1.1911j            1.2605           -0.1970
AR.13            0.4124           +1.1911j            1.2605            0.1970
MA.1             1.0000           -0.0000j            1.0000           -0.0000
MA.2            -0.8859           -0.4642j            1.0001           -0.4232
MA.3            -0.8859           +0.4642j            1.0001            0.4232
-----------------------------------------------------------------------------
```

Рисунок 7 – Таблица коэффициентов

Найдем постоянное среднее и дисперсию.

```
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Отклонения", ax=ax[0])
residuals.plot(kind='kde', title='Плотность', ax=ax[1])
plt.show()
```

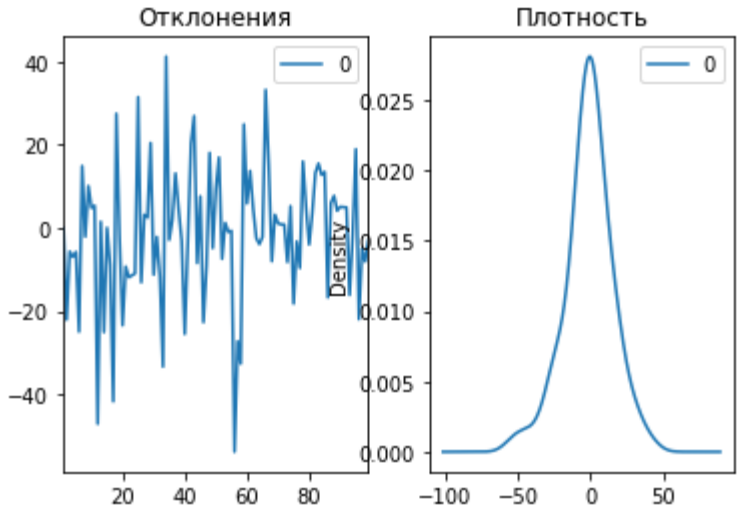


Рисунок 8 – Графики постоянного среднего и дисперсии

Остаточные ошибки кажутся хорошими с почти нулевым средним и равномерной дисперсией. Следует построить фактические данные в соответствии с установленными значениями, используя «plot predict».

```
model_fit.plot_predict(dynamic=False)
plt.show()
```

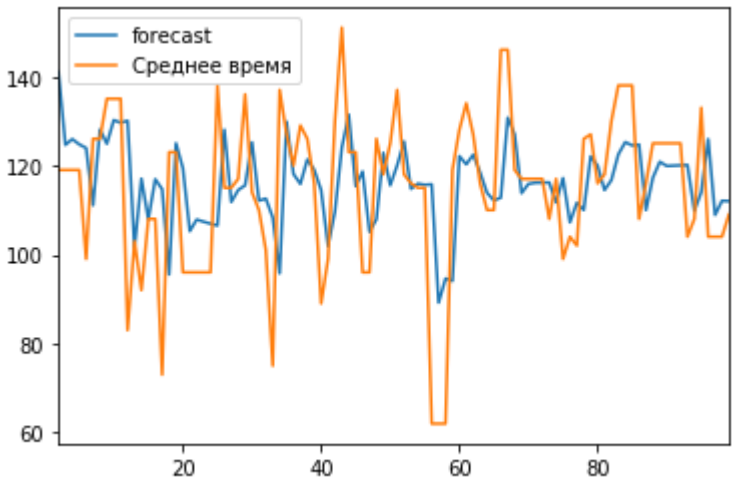


Рисунок 9 – График фактических и приспособленных значений

При установке в выборке запаздывающие значения используются для прогнозирования «dynamic=False»

То есть модель обучается до предыдущего значения, чтобы сделать следующий прогноз. Это помогает сделать подходящий прогноз, однако фактические данные выглядят искусственно.

```
train1 = X[:80]
test1 = X[80:]
```

```
model = ARIMA(train1, order=(13, 1, 3))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(20, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test1.index)
lower_series = pd.Series(conf[:, 0], index=test1.index)
upper_series = pd.Series(conf[:, 1], index=test1.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train1, label='Тренировка')
plt.plot(test1, label='Реальность')
plt.plot(fc_series, label='Предсказание')
plt.title('Предсказание vs Реальности')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

Рисунок 10 – График предсказанных, реальных и «натреннированных» значений

Аналогично проделываем для прогнозирования количества машин и средней скорости.

Прогнозирование количества машин:

```
Y = new_traffic['Кол-во машин']

# Original Series
fig, axes = plt.subplots(3, 2)
axes[0, 0].plot(Y); axes[0, 0].set_title('Изначальные данные')
plot_acf(Y, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(Y.diff()); axes[1, 0].set_title('Дифференциирование 1-ого порядка')
plot_acf(Y.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(Y.diff().diff()); axes[2, 0].set_title('Дифференциирование 2-ого порядка')
plot_acf(Y.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```

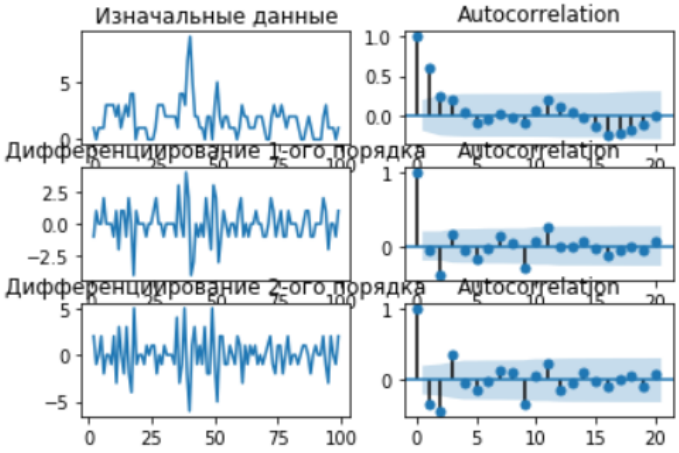


Рисунок 11 – Задание начальных данных и данных для дифференцирования

```
model = ARIMA(Y, order=(13,1,4))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:          D.Кол-во машин   No. Observations:                   99
Model:                ARIMA(13, 1, 4)   Log Likelihood                -143.706
Method:                       css-mle   S.D. of innovations              0.998
Date:                Fri, 12 Jul 2019   AIC                            325.412
Time:                        17:58:49   BIC                            374.719
Sample:                             1   HQIC                           345.361

================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           -0.0041      0.036     -0.113      0.911      -0.075       0.067
ar.L1.D.Кол-во машин            -0.2914      0.128     -2.275      0.026      -0.543      -0.040
ar.L2.D.Кол-во машин            -0.1067      0.155     -0.688      0.494      -0.411       0.198
ar.L3.D.Кол-во машин            -0.2940      0.159     -1.853      0.067      -0.605       0.017
ar.L4.D.Кол-во машин            -1.1067      0.128     -8.625      0.000      -1.358      -0.855
ar.L5.D.Кол-во машин            -0.4588      0.158     -2.902      0.005      -0.769      -0.149
ar.L6.D.Кол-во машин            -0.5854      0.154     -3.797      0.000      -0.888      -0.283
ar.L7.D.Кол-во машин            -0.1887      0.163     -1.160      0.249      -0.508       0.130
ar.L8.D.Кол-во машин            -0.4324      0.149     -2.898      0.005      -0.725      -0.140
ar.L9.D.Кол-во машин            -0.5428      0.144     -3.763      0.000      -0.825      -0.260
ar.L10.D.Кол-во машин           -0.2936      0.119     -2.459      0.016      -0.528      -0.060
ar.L11.D.Кол-во машин           -0.0281      0.120     -0.233      0.816      -0.264       0.208
ar.L12.D.Кол-во машин           -0.1021      0.114     -0.897      0.372      -0.325       0.121
ar.L13.D.Кол-во машин           -0.2933      0.113     -2.588      0.011      -0.515      -0.071
ma.L1.D.Кол-во машин             0.2291      0.081      2.825      0.006       0.070       0.388
ma.L2.D.Кол-во машин            -0.4734      0.117     -4.055      0.000      -0.702      -0.245
ma.L3.D.Кол-во машин             0.2291      0.064      3.551      0.001       0.103       0.355
ma.L4.D.Кол-во машин             1.0000        nan        nan        nan         nan         nan
                                    Roots
==============================================================================
                  Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1            -0.8432           -0.5476j            1.0054           -0.4083
AR.2            -0.8432           +0.5476j            1.0054            0.4083
AR.3            -1.2484           -0.0000j            1.2484           -0.5000
AR.4            -0.7746           -0.8092j            1.1202           -0.3715
AR.5            -0.7746           +0.8092j            1.1202            0.3715
AR.6            -0.2102           -1.0668j            1.0873           -0.2810
AR.7            -0.2102           +1.0668j            1.0873            0.2810
AR.8             0.9446           -0.5809j            1.1089           -0.0877
AR.9             0.9446           +0.5809j            1.1089            0.0877
AR.10            0.7611           -0.7247j            1.0510           -0.1211
AR.11            0.7611           +0.7247j            1.0510            0.1211
AR.12            0.5724           -1.0067j            1.1580           -0.1677
AR.13            0.5724           +1.0067j            1.1580            0.1677
MA.1             0.7312           -0.6822j            1.0000           -0.1195
MA.2             0.7312           +0.6822j            1.0000            0.1195
MA.3            -0.8457           -0.5337j            1.0000           -0.4104
MA.4            -0.8457           +0.5337j            1.0000            0.4104
------------------------------------------------------------------------------
```

Рисунок 12 – Таблица коэффициентов

```
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Отклонения", ax=ax[0])
residuals.plot(kind='kde', title='Плотность', ax=ax[1])
plt.show()
```

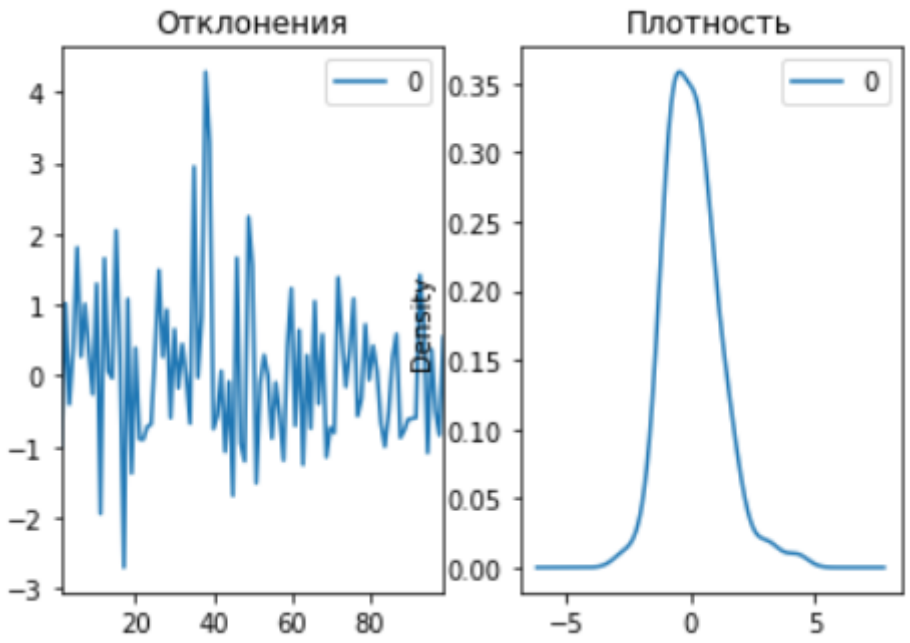


Рисунок 13 – Графики постоянного среднего и дисперсии

```
model_fit.plot_predict(dynamic=False)
plt.show()
```

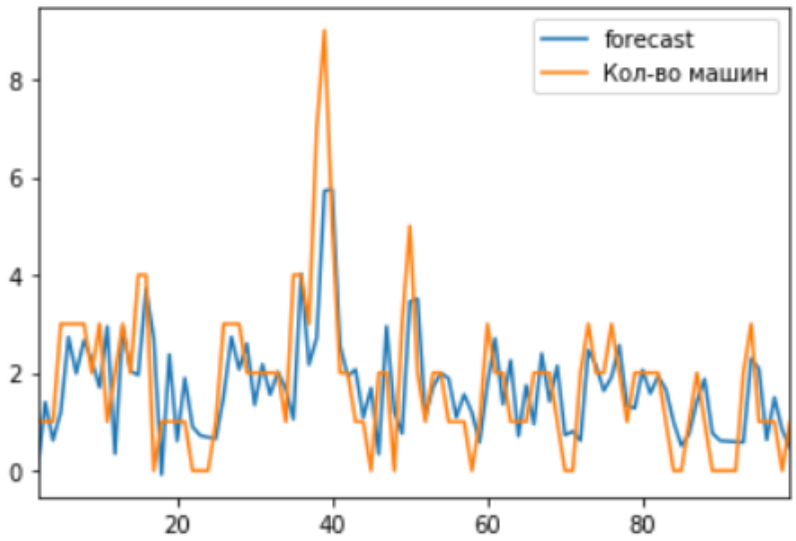


Рисунок 14 – График фактических и приспособленных значений

```
train2 = Y[:80]
test2 = Y[80:]

model = ARIMA(train2, order=(13, 1, 4))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(20, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test2.index)
lower_series = pd.Series(conf[:, 0], index=test2.index)
upper_series = pd.Series(conf[:, 1], index=test2.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train2, label='Тренировка')
plt.plot(test2, label='Реальность')
plt.plot(fc_series, label='Предсказание')
plt.title('Предсказание vs Реальности')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

Рисунок 15 – График предсказанных, реальных и «натреннированных» значений

Прогнозирование средней скорости:

```
Z = new_traffic['Средняя скорость']

# Original Series
fig, axes = plt.subplots(3, 2)
axes[0, 0].plot(Z); axes[0, 0].set_title('Изначальные данные')
plot_acf(Z, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(Z.diff()); axes[1, 0].set_title('Дифференциирование 1-ого порядка')
plot_acf(Z.diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(Z.diff().diff()); axes[2, 0].set_title('Дифференциирование 2-ого порядка')
plot_acf(Z.diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```

```
model = ARIMA(Z, order=(7,1,2))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:     D.Средняя скорость   No. Observations:                   99
Model:                 ARIMA(7, 1, 2)   Log Likelihood                -384.721
Method:                       css-mle   S.D. of innovations             11.548
Date:                Fri, 12 Jul 2019   AIC                            791.442
Time:                        17:59:45   BIC                            819.988
Sample:                             1   HQIC                           802.992

================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           -0.0261      0.064     -0.406      0.686      -0.152       0.100
ar.L1.D.Средняя скорость        -0.2121      0.331     -0.641      0.523      -0.861       0.437
ar.L2.D.Средняя скорость         0.3229      0.185      1.742      0.085      -0.040       0.686
ar.L3.D.Средняя скорость        -0.1178      0.108     -1.089      0.279      -0.330       0.094
ar.L4.D.Средняя скорость        -0.1085      0.115     -0.945      0.347      -0.334       0.117
ar.L5.D.Средняя скорость         0.0047      0.108      0.044      0.965      -0.207       0.217
ar.L6.D.Средняя скорость        -0.0278      0.107     -0.261      0.795      -0.237       0.181
ar.L7.D.Средняя скорость         0.0735      0.108      0.678      0.499      -0.139       0.286
ma.L1.D.Средняя скорость        -0.2881      0.320     -0.900      0.370      -0.915       0.339
ma.L2.D.Средняя скорость        -0.7119      0.319     -2.230      0.028      -1.338      -0.086
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.1742           -0.4158j            1.2456           -0.4458
AR.2           -1.1742           +0.4158j            1.2456            0.4458
AR.3           -0.4122           -1.6094j            1.6614           -0.2899
AR.4           -0.4122           +1.6094j            1.6614            0.2899
AR.5            0.9617           -1.0133j            1.3970           -0.1292
AR.6            0.9617           +1.0133j            1.3970            0.1292
AR.7            1.6270           -0.0000j            1.6270           -0.0000
MA.1            1.0000           +0.0000j            1.0000            0.0000
MA.2           -1.4046           +0.0000j            1.4046            0.5000
-----------------------------------------------------------------------------
```

```
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Отклонения", ax=ax[0])
residuals.plot(kind='kde', title='Плотность', ax=ax[1])
plt.show()
```

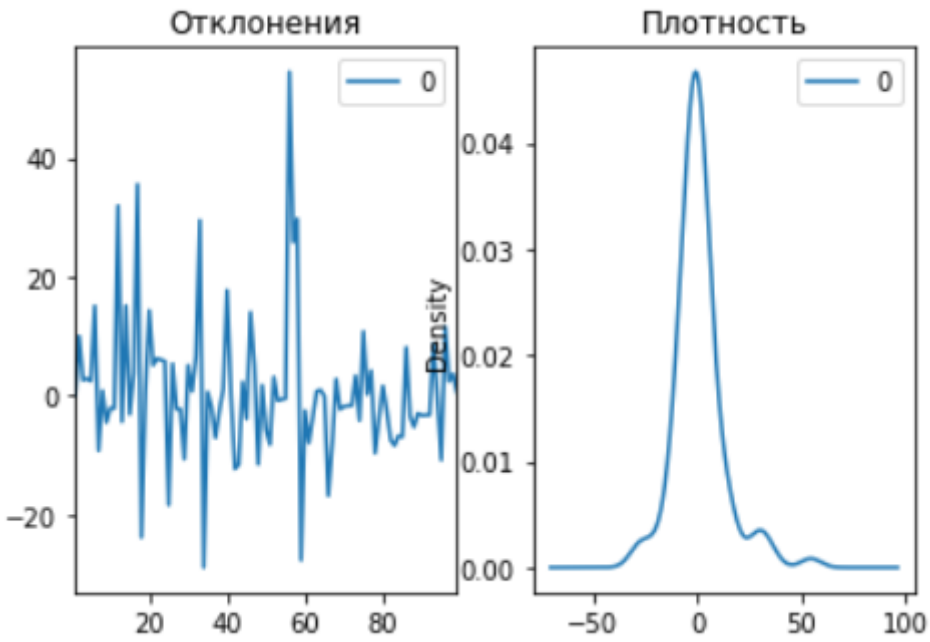


Рисунок 16 – Задание начальных данных и данных для дифференцирования

```
model_fit.plot_predict(dynamic=False)
plt.show()
```

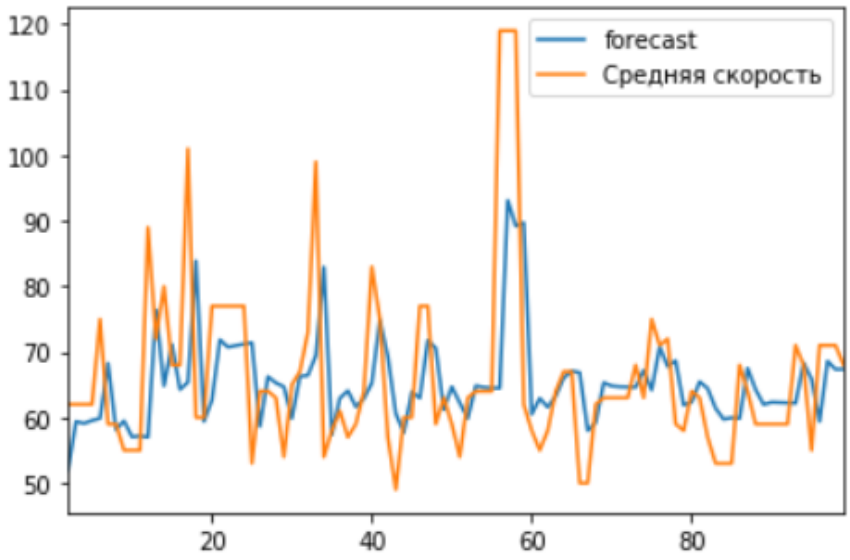


Рисунок 17 – График фактических и приспособленных значений

```
train3 = Z[:80]
test3 = Z[80:]
```

```
model = ARIMA(train3, order=(7, 1,2))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(20, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test3.index)
lower_series = pd.Series(conf[:, 0], index=test3.index)
upper_series = pd.Series(conf[:, 1], index=test3.index)

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train3, label='Тренировка')
plt.plot(test3, label='Реальность')
plt.plot(fc_series, label='Предсказание')
plt.title('Предсказание vs Реальности')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

Рисунок 18 – График предсказанных, реальных и «натреннированных» значений

Модель, разработанная с использованием алгоритма Sarima

Построим модель SARIMA, используя «pmdarima S».

Прогнозирование среднего времени:

```
smodel = pm.auto_arima(X, start_p=1, start_q=1,test='adf',max_p=3, max_q=3, m=52,start_P=0, seasonal=True,
                       d=None, D=1, trace=True,error_action='ignore',  suppress_warnings=True, stepwise=True)
```

```
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=447.391, BIC=451.091, Fit time=0.289 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=444.980, BIC=450.531, Fit time=0.551 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=438.020, BIC=445.421, Fit time=1.736 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=440.774, BIC=451.875, Fit time=1.682 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=440.128, BIC=445.679, Fit time=0.590 seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=439.991, BIC=449.241, Fit time=3.104 seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=440.589, BIC=449.840, Fit time=1.363 seconds
Total fit time: 9.341 seconds
```

Рисунок 19 – Подбор коэффициентов

```
print(smodel.summary())
```

```
                                Statespace Model Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  100
Model:             SARIMAX(1, 1, 1)x(0, 1, 0, 52)   Log Likelihood                -215.010
Date:                            Thu, 11 Jul 2019   AIC                            438.020
Time:                                    14:39:36   BIC                            445.421
Sample:                                         0   HQIC                           440.805
                                              - 100
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.2263      0.425      0.533      0.594      -0.606       1.059
ar.L1          0.4282      0.231      1.856      0.063      -0.024       0.880
ma.L1         -0.9995     15.573     -0.064      0.949     -31.522      29.523
sigma2       517.2061   8017.730      0.065      0.949   -1.52e+04    1.62e+04
===================================================================================
Ljung-Box (Q):                       29.72   Jarque-Bera (JB):                 1.11
Prob(Q):                              0.88   Prob(JB):                         0.58
Heteroskedasticity (H):               1.01   Skew:                            -0.25
Prob(H) (two-sided):                  0.99   Kurtosis:                         3.56
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Рисунок 20 – Таблица данных

```
predict = smodel.fit_predict(train1, n_periods=20)
```

```
predict_series = pd.Series(predict, index=test1.index)
```

```
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train1, label='training')
plt.plot(test1, label='actual')
plt.plot(predict_series, label='predict SARIMA')
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

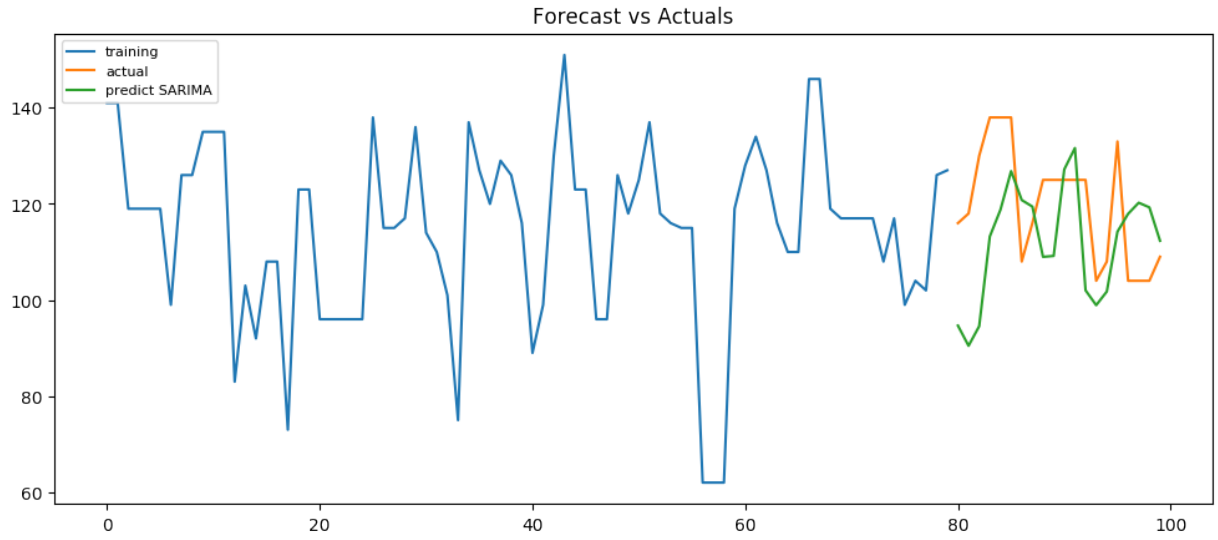


Рисунок 21 – График предсказанных, реальных и «натреннированных» значений

Аналогично проделываем для прогнозирования количества машин и средней скорости.

Прогнозирование количества машин:

```
smodel = pm.auto_arima(Y, start_p=1, start_q=1,test='adf',max_p=3, max_q=3, m=52,start_P=0, seasonal=True,
                       d=None, D=1, trace=True,error_action='ignore',  suppress_warnings=True, stepwise=True)
```

```
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=190.967, BIC=194.667, Fit time=0.213 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=192.791, BIC=198.341, Fit time=0.304 seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=192.325, BIC=197.876, Fit time=0.399 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=186.984, BIC=194.385, Fit time=0.675 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=184.761, BIC=194.011, Fit time=0.948 seconds
Fit ARIMA: order=(2, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=188.807, BIC=196.207, Fit time=0.484 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=181.171, BIC=192.272, Fit time=4.493 seconds
Fit ARIMA: order=(3, 1, 3) seasonal_order=(0, 1, 0, 52); AIC=182.949, BIC=197.751, Fit time=4.938 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=179.737, BIC=188.988, Fit time=2.436 seconds
Fit ARIMA: order=(1, 1, 3) seasonal_order=(0, 1, 0, 52); AIC=181.781, BIC=192.882, Fit time=4.749 seconds
Fit ARIMA: order=(2, 1, 3) seasonal_order=(0, 1, 0, 52); AIC=183.076, BIC=196.027, Fit time=4.438 seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=180.553, BIC=187.954, Fit time=0.694 seconds
Total fit time: 24.815 seconds
```

Рисунок 22 – Подбор коэффициентов

```
print(smodel.summary())
```

```
                                Statespace Model Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  100
Model:             SARIMAX(1, 1, 2)x(0, 1, 0, 52)   Log Likelihood                -84.869
Date:                            Thu, 11 Jul 2019   AIC                            179.737
Time:                                    14:40:37   BIC                            188.988
Sample:                                         0   HQIC                           183.218
                                              - 100
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.0221      0.028     -0.782      0.434      -0.077       0.033
ar.L1          0.2570      0.198      1.295      0.195      -0.132       0.646
ma.L1         -0.1317      9.780     -0.013      0.989     -19.301      19.037
ma.L2         -0.8674      8.459     -0.103      0.918     -17.446      15.711
sigma2         1.9946     19.362      0.103      0.918     -35.954      39.943
===================================================================================
Ljung-Box (Q):                       41.78   Jarque-Bera (JB):                 2.84
Prob(Q):                              0.39   Prob(JB):                         0.24
Heteroskedasticity (H):               2.70   Skew:                            -0.60
Prob(H) (two-sided):                  0.06   Kurtosis:                         2.96
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Рисунок 23 – Таблица данных

```
predict = smodel.fit_predict(train2, n_periods=20)
```

```
predict_series = pd.Series(predict, index=test2.index)
```

```
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train2, label='training')
plt.plot(test2, label='actual')
plt.plot(predict_series, label='predict SARIMA')
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

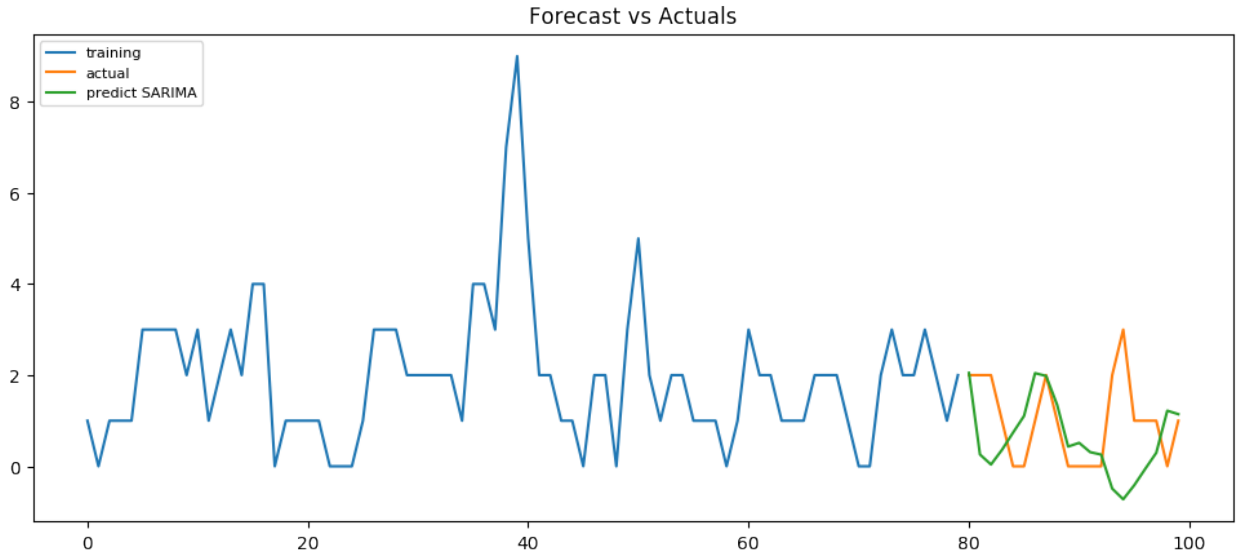


Рисунок 24 – График предсказанных, реальных и «натреннированных» значений

Прогнозирование средней скорость:

```
smodel = pm.auto_arima(Z, start_p=1, start_q=1,test='adf',max_p=3, max_q=3, m=52,start_P=0, seasonal=True,
                       d=None, D=1, trace=True,error_action='ignore',  suppress_warnings=True, stepwise=True)
```

```
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=416.556, BIC=420.257, Fit time=0.269 seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(0, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 0) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 0) seasonal_order=(0, 1, 0, 52); AIC=414.764, BIC=420.314, Fit time=0.492 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=408.178, BIC=415.579, Fit time=2.101 seconds
Fit ARIMA: order=(2, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=408.242, BIC=419.343, Fit time=2.643 seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 0, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(1, 1, 1) seasonal_order=(1, 1, 1, 52); AIC=nan, BIC=nan, Fit time=nan seconds
Fit ARIMA: order=(0, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=411.452, BIC=417.003, Fit time=0.653 seconds
Fit ARIMA: order=(2, 1, 1) seasonal_order=(0, 1, 0, 52); AIC=410.151, BIC=419.402, Fit time=2.240 seconds
Fit ARIMA: order=(1, 1, 2) seasonal_order=(0, 1, 0, 52); AIC=408.495, BIC=417.745, Fit time=2.504 seconds
Total fit time: 10.935 seconds
```

Рисунок 25 – Подбор коэффициентов

```
print(smodel.summary())
```

```
                                Statespace Model Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                  100
Model:             SARIMAX(1, 1, 1)x(0, 1, 0, 52)   Log Likelihood                -200.089
Date:                            Thu, 11 Jul 2019   AIC                            408.178
Time:                                    14:41:22   BIC                            415.579
Sample:                                         0   HQIC                           410.963
                                              - 100
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.1926      0.343     -0.561      0.575      -0.865       0.480
ar.L1          0.4645      0.236      1.966      0.049       0.002       0.928
ma.L1         -0.9998     52.464     -0.019      0.985    -103.827     101.828
sigma2       274.5365   1.44e+04      0.019      0.985   -2.79e+04    2.84e+04
==============================================================================
Ljung-Box (Q):                       33.69   Jarque-Bera (JB):                 3.95
Prob(Q):                              0.75   Prob(JB):                         0.14
Heteroskedasticity (H):               0.65   Skew:                             0.34
Prob(H) (two-sided):                  0.40   Kurtosis:                         4.25
==============================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Рисунок 26 – Таблица данных

```
predict = smodel.fit_predict(train3, n_periods=20)
```

```
predict_series = pd.Series(predict, index=test3.index)
```

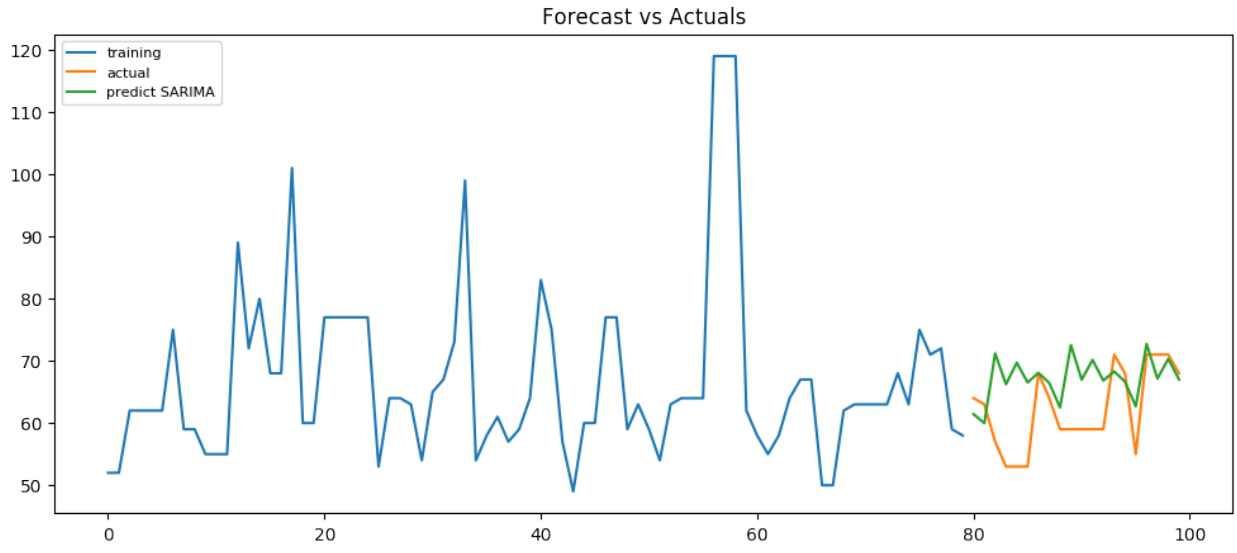


Рисунок 27 – График предсказанных, реальных и «натреннированных» значений

3.3. Вывод

По итогам выполнения работы был разработан и реализован компонент информационной системы, написаный на языке программирования Python в среде разработки Jupyter Notebook, который способен с некоторой точностью прогнозировать характеристики транспортных потоков.

Осуществив анализ полученных данных при помощи функции «forecast_accuracy» были получены следующие результаты средней абсолютной ошибки в процентах:

1) Среднее время: 0.12311452317359892
2) Количество машин: 0.17342123432212749
3) Средняя скорость: 0.1121308410725749

Исходя из чего можно сделать вывод о том, что значения среднего времени были предсказаны с вероятностью 87%, количества машин с вероятностью 83% и средней скорости с вероятностью 89%.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
2. machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
3. cyberleninka.ru/article/v/metodika-postroeniya-modeli-arima-dlya-prognozirovaniya-dinamiki-vremennyh-ryadov
4. github.com/rafat/ctsa/wiki/SARIMA
5. wikipedia.org/wiki/Autoregressive_integrated_moving_average
6. github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/23_Time-Series-Prediction.ipynb