Тема: **Прогноз розничных продаж на Python.**

Цель: научиться понимать методы исследовательского анализа данных с помощью языка программирования Python.

Материальное и дидактическое оснащение: Методические рекомендации по выполнению практической работы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Анализ временных рядов

Временной ряд — это ряд точек данных, проиндексированных (или перечисленных, или нарисованных на графике) во временном порядке. Анализ временных рядов относится к методам извлечения значимой статистики из данных временных рядов. Это обычно используется для прогнозирования и других моделей.Первоисточник

Цели обучения

Понимание использования анализа временных рядов плюсы и минусы различных методов TSA, включая различие между линейными и нелинейными методами. Проанализируйте результаты на заданных данных магазина rossman.

Описание проблемы

Для этого ноутбука мы будем использовать базу данных продаж магазина rossman. Ниже приводится описание данных с веб-сайта:

«Rossmann управляет более чем 3000 аптеками в 7 европейских странах. В настоящее время перед менеджерами магазинов Rossmann стоит задача прогнозировать свои ежедневные продажи на срок до шести недель вперед. На продажи в магазинах влияет множество факторов, включая рекламные акции, конкуренция, школьные и государственные праздники, сезонность и местоположение. Поскольку тысячи отдельных менеджеров прогнозируют продажи на основе своих уникальных обстоятельств, точность результатов может быть весьма различной».

ПРАКТИЧЕСКАЯ ЧАСТЬ РАБОТЫ

В ходе изучения теоретических материалов теоретических сведений данной работы:

I. Ознакомьтесь с теоретическими сведениями.

II. Изучите инструкцию ниже.

III. Предоставьте ссылку на свою работу в https://colab.research.google.com/ онлайн, в процессе выполнения преподавателю на yvmranh48@gmail.com, через вкладку Share (поделиться). Для отслеживания Вашей работы преподавателем онлайн и возможности консультирования

IV. Оформите работу, используя снимки экрана.

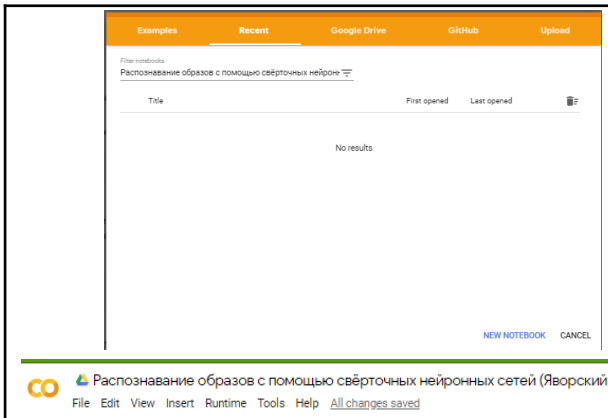
V. Ссылку на работу и отчет по работе выложите в ментальной карте.

VI. Сделайте алгоритм по Вашей работе в draw.io

VII. Выкладывайте отчет на странице своего сайта.

Ход работы.

Во время работы не забывайте запускать на каждом этапе код.

<table>
<tr><td rowspan="2"> 
</td><td>Создайте новый ноутбук по ссылке: https://colab.research.google.com/.</td></tr>
<tr><td>Назовите:
Исследовательский анализ данных на Python (Ф.И.О.).ipynb.
Подпишите в скобках своим Ф.И.О.</td></tr>
<tr><td colspan="2">Ваш новый ноутбук должен содержать комментарии и иметь законченный вид.
Текст для комментариев можем использовать из теоретических сведений.</td></tr>
<tr><td>Импортируем необходимые библиотеки</td><td><pre>
# Library Imports
import numpy as np
import pandas as pd
import matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
from scipy.stats import skew
from scipy.stats.stats import pearso
nr
from math import sqrt
from sklearn.metrics import mean_squ
ared_error

matplotlib parameters
matplotlib.rcParams['axes.labelsize'
] = 14
matplotlib.rcParams['xtick.labelsize
'] = 12
matplotlib.rcParams['ytick.labelsize
'] = 12
matplotlib.rcParams['text.color'] =
'k'

%config InlineBackend.figure_format
= 'retina'
%matplotlib inline
</pre></td></tr>
<tr><td>Чтение данных</td><td><pre>
# Data Reading
train = pd.read_csv('https://raw.githubuse
rcontent.com/RPI-DATA/tutorials-intro/mast
er/rossmann-store-sales/rossmann-store-sal
es/train.csv', parse_dates = True, low_mem
ory = False, index_col = 'Date')
</pre></td></tr>
</table>

	`store = pd.read_csv('https://raw.githubusercontent.com/RPI-DATA/tutorials-intro/master/rossmann-store-sales/rossmann-store-sales/store.csv', low_memory = False)`

Исследовательский анализ данных
Начнем с просмотра того, из чего состоят наши данные. Мы хотим увидеть, какие переменные являются непрерывными, а какие — категориальными. Изучив некоторые данные, мы видим, что можем создать функцию. Количество продаж, разделенное на клиентов, может дать нам хороший показатель для измерения средних продаж на одного клиента. Мы также можем сделать предположение, что если в этом столбце отсутствуют значения, то у нас 0 клиентов. Поскольку клиенты стимулируют продажи, мы решили удалить все эти значения.
 	
Обратите внимание на порядок, в котором перечислены данные. Он упорядочен от самой последней даты до самой старой даты. Это может вызвать проблемы при разработке нашей модели.
	`train.head()`
	Store DayOfWeek Sales Customers Open Promo StateHoliday SchoolHoliday
	Date
	2015-07-31 1 5 5263 555 1 1 0 1
	2015-07-31 2 5 6064 625 1 1 0 1
	2015-07-31 3 5 8314 821 1 1 0 1
	2015-07-31 4 5 13995 1498 1 1 0 1
	2015-07-31 5 5 4822 559 1 1 0 1
	
	`train.shape`
	`(1017209, 8)`

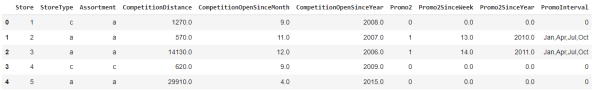
После этого мы будем использовать удивительную. describe()функцию, которая может предоставить нам большинство статистических элементов.	`train.describe()` Store DayOfWeek Sales Customers Open Promo SchoolHoliday count 1.017209e+06 1.017209e+06 1.017209e+06 1.017209e+06 1.017209e+06 1.017209e+06 1.017209e+06 mean 5.584297e+02 3.998341e+00 5.773819e+03 6.331459e+02 8.301067e-01 3.815145e-01 1.786467e-01 std 3.219087e+02 1.997391e+00 3.849926e+03 4.644117e+02 3.755392e-01 4.857586e-01 3.830564e-01 min 1.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 25% 2.800000e+02 2.000000e+00 3.727000e+03 4.050000e+02 1.000000e+00 0.000000e+00 0.000000e+00 50% 5.580000e+02 4.000000e+00 5.744000e+03 6.090000e+02 1.000000e+00 0.000000e+00 0.000000e+00 75% 8.380000e+02 6.000000e+00 7.856000e+03 8.370000e+02 1.000000e+00 1.000000e+00 0.000000e+00 max 1.115000e+03 7.000000e+00 4.155100e+04 7.388000e+03 1.000000e+00 1.000000e+00 1.000000e+00

Мы проверим все недостающие элементы здесь.	`missing = train.isnull().sum()`
	`missing.sort_values(ascending=False)`
	Store 0
	DayOfWeek 0
	Sales 0
	Customers 0
	Open 0
	Promo 0
	StateHoliday 0
	SchoolHoliday 0
	dtype: int64

Затем мы создаем новую метрику, чтобы увидеть средние продажи на одного клиента.	`train['SalesPerCustomer'] = train['Sales'] /train['Customers']` `train['SalesPerCustomer'].head()`

Мы собираемся проверить, есть ли какие-либо пропущенные значения с нашей новой метрикой, и удалить их. Либо клиенты, либо продажи должны быть равны нулю, чтобы дать нам нулевое значение SalesPerCustomer.	`missing = train.isnull().sum()`
	`missing.sort_values(ascending=False)`
	SalesPerCustomer 172869
	Store 0
	DayOfWeek 0
	Sales 0
	Customers 0
	Open 0
	Promo 0
	StateHoliday 0
	SchoolHoliday 0
	dtype: int64

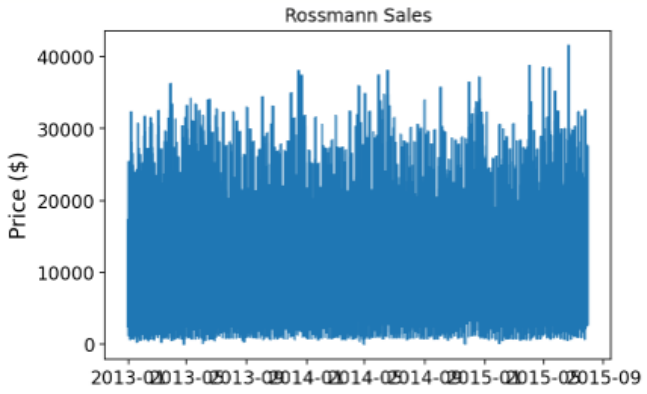
	`train.dropna().head()` Store DayOfWeek Sales Customers Open Promo StateHoliday SchoolHoliday SalesPerCustomer Date 2015-07-31 1 5 5263 555 1 1 0 1 9.482883 2015-07-31 2 5 6064 625 1 1 0 1 9.702400 2015-07-31 3 5 8314 821 1 1 0 1 10.126675 2015-07-31 4 5 13995 1498 1 1 0 1 9.342457 2015-07-31 5 5 4822 559 1 1 0 1 8.626118
Исследовательский анализ данных (сохранение данных) Мы делаем то же самое, что и для нашего тренировочного набора. Изучая данные, мы видим, что в CompetitionDistance отсутствуют только 3 значения. Поскольку это такое небольшое количество, мы решили заменить их средним значением столбца. Остальные отсутствующие значения зависят от Promo2. Поскольку эти отсутствующие значения связаны с тем, что Promo2 равен 0, мы можем заменить эти нули на 0.	`store.head()` Store StoreType Assortment CompetitionDistance CompetitionOpenSinceMonth CompetitionOpenSinceYear Promo2 Promo2SinceWeek Promo2SinceYear PromoInterval 0 1 c a 1270.0 9.0 2008.0 0 NaN NaN NaN 1 2 a a 570.0 11.0 2007.0 1 13.0 2010.0 Jan,Apr,Jul,Oct 2 3 a a 14130.0 12.0 2006.0 1 14.0 2011.0 Jan,Apr,Jul,Oct 3 4 c c 620.0 9.0 2009.0 0 NaN NaN NaN 4 5 a a 29910.0 4.0 2015.0 0 NaN NaN NaN
	`store.shape` (1115, 10)
	`store.isnull().sum()` Store 0 StoreType 0 Assortment 0 CompetitionDistance 3 CompetitionOpenSinceMonth 354 CompetitionOpenSinceYear 354 Promo2 0 Promo2SinceWeek 544 Promo2SinceYear 544 PromoInterval 544 dtype: int64
Так как из этого пропущено только 3 значения, мы заполняем средним значением из столбца	`store['CompetitionDistance'].fillna(store['CompetitionDistance'].mean(), inplace = True)`

Строки, в которых нет Promo2, мы можем заполнить остальные значения 0	`store.fillna(0, inplace = True)`
	`store.head()`
	 

Строки, в которых нет Promo2, мы можем заполнить остальные значения 0	`train = train.merge(right=store, on='Store', how='left')`
Модель скользящего среднего (Наивная модель)	

Мы собираемся использовать модель скользящего среднего для прогнозирования акций GM для нашей базовой модели. Модель скользящего среднего будет использовать среднее значение различных «окон» времени, чтобы составить свой прогноз.
 	
Мы перезагружаем данные, потому что теперь у нас есть представление о том, как мы хотим манипулировать ими для нашей модели. Проделав те же манипуляции с данными, что и раньше, мы начинаем смотреть на тенденцию наших продаж.
	`train = pd.read_csv("https://raw.githubusercontent.com/RPI-DATA/tutorials-intro/master/rossmann-store-sales/rossmann-store-sales/train.csv", parse_dates = True, low_memory = False, index_col = 'Date')`
	`train = train.sort_index(ascending = True)`
	
	`train['SalesPerCustomer'] = train['Sales']/train['Customers']`
	`train['SalesPerCustomer'].head()`
	`Date`
	`2013-01-01  3.828049`
	`2013-01-01  4.981328`
	`2013-01-01  4.233600`
	`2013-01-01  5.907021`
	`2013-01-01  5.464286`
	`Name: SalesPerCustomer, dtype: float64`

	`train = train.dropna()`

Здесь мы просто строим график продаж, которые у нас есть. Как видите, количество продаж настолько велико, что наш график просто выглядит как голубая заливка. Однако мы можем понять, как распределяются наши продажи.	`plt.plot(train.index, train['Sales'])`
	`plt.title('Rossmann Sales')`
	`plt.ylabel('Price ($)');`
	`plt.show()`
	 

Чтобы очистить наш график, мы хотим сформировать новый столбец, учитывающий только год продаж.	`train['Year'] = train.index.year`
	
	`# Take Dates from index and move to Date column`
	`train.reset_index(level=0, inplace = True)`
	`train['sales'] = 0`

Разделите данные на набор решений и тестов. Мы используем	`train_store=train[0:675472]` `test_store=train[675472:]`

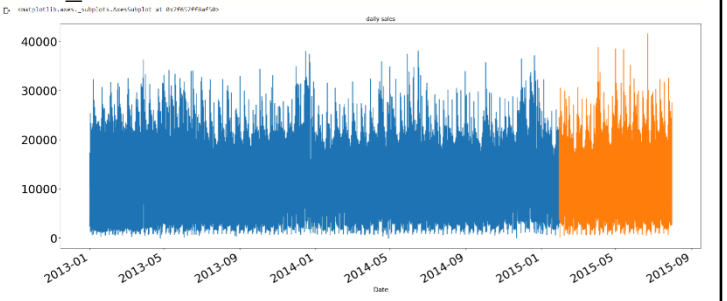
разделение 80/20. Затем мы смотрим, чтобы начать модеин.

test_store означает часть прогнозирования.

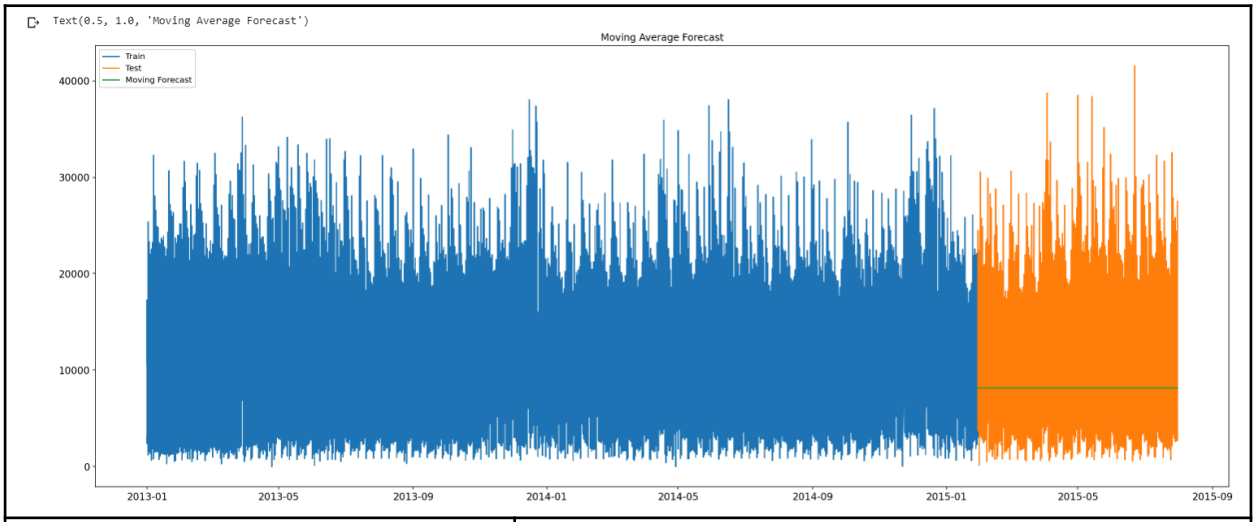
```
train_store.Date = pd.to_datetime(train_store.Date, format="%Y-%m-%d")
train_store.index = train_store.Date
test_store.Date = pd.to_datetime(test_store.Date, format="%Y-%m-%d")
test_store.index = test_store.Date

train_store = train_store.resample('D').mean()
train_store = train_store.interpolate(method='linear')

test_store = test_store.resample('D').mean()
test_store = test_store.interpolate(method='linear')
```

```
/Users/josefigueroa/anaconda3/lib/python3.7/site-packages/pandas/core/generic.py:4405: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-c
  self[name] = value
```

```
train_store.Sales.plot(figsize=(25,10), title='daily sales', fontsize=25)
test_store.Sales.plot()
```




```
y_hat_avg_moving = test_store.copy()
y_hat_avg_moving['moving_avg_forcast'] = train_store.Sales.rolling(90).mean().iloc[-1]
plt.figure(figsize=(25,10))
plt.plot(train_store['Sales'], label='Train')
plt.plot(test_store['Sales'], label='Test')
plt.plot(y_hat_avg_moving['moving_avg_forcast'], label='Moving Forecast')
plt.legend(loc='best')
plt.title('Moving Average Forecast')
```




```
rms_avg_rolling = sqrt(mean_squared_error(
test_store.Sales, y_hat_avg_moving.moving_
avg_forcast))
print('ROLLING AVERAGE',rms_avg_rolling)
```

```
ROLLING AVERAGE 3233.627446795865
```

Скользящее среднее для нашей модели составляет 1915,88. Этот прогноз кажется очень последовательным в достижении среднего значения будущих продаж. Эта наивная модель определенно выглядит солидной моделью, однако она не самая лучшая.