ДОДАТОК А

Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

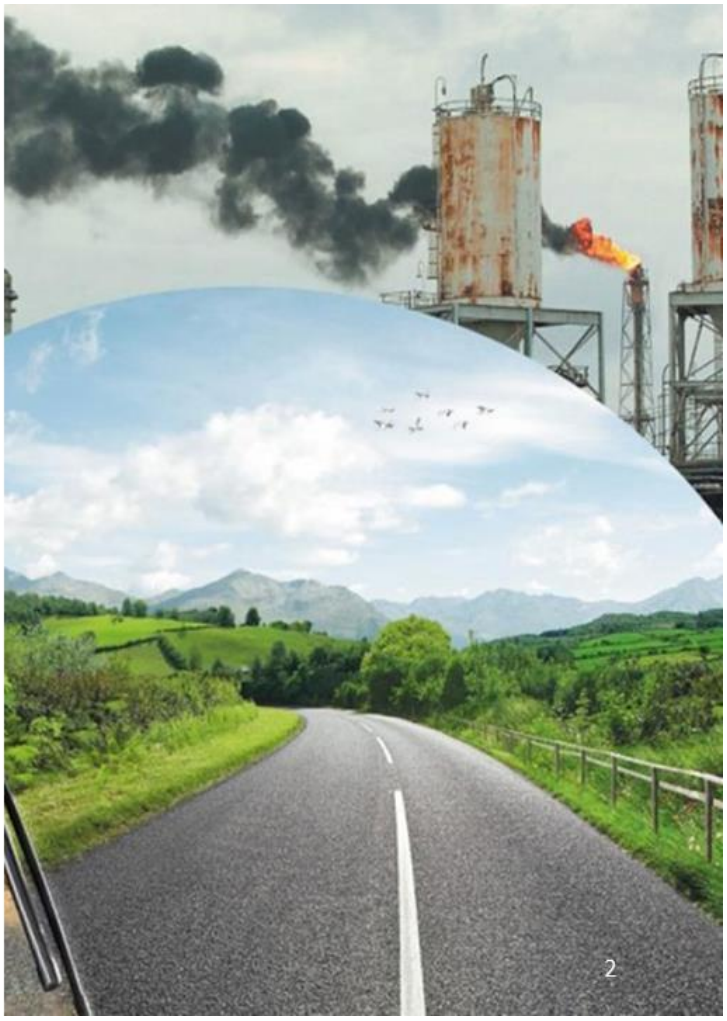
Дослідження авторегресійних й нейромережевих моделей для короткострокового прогнозування забруднення повітря

Виконала:
ст.гр. ІПЗм-18-2
Таламанова І.С.

Керівник:
проф. Смеляков К.С.

Актуальність теми

Забруднення повітря має великий вплив на здоров'я та якість життя людей. Для людей із **захворюваннями серця та легенів** таке забруднення дуже небезпечне.



Мета роботи

Метою роботи є дослідження найефективніших методів короткострокового передбачення забруднення повітря, а також дослідження застосування цих методів для практичних задач.

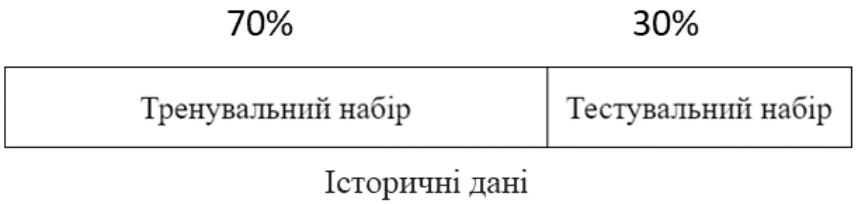
Задачею дослідження є відповісти на питання:

Які з методів, авторегресійні або нейромережеві, є найбільш ефективними для передбачення забруднення повітря?

Забруднення повітря

- Типи забрудників:

  CO, NO2, SO2, O3, ТЧ2.5, ТЧ10.
- Тверді частинки (ТЧ) один з **найнебезпечніших забрудників**.
- Забруднення повітря це **часовий ряд**.
- Часовий ряд може мати паттерни: тренд, сезонність, цикли.

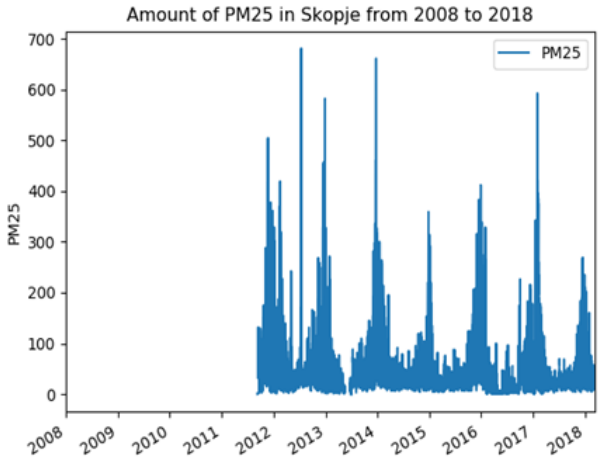


Набір даних

Проблема:
Зашумленість даних

Рішення:

- Усунути викиди
- Трансформація Бокса-Кокса



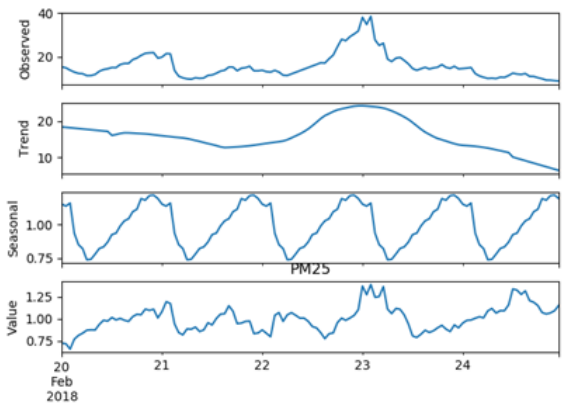
“Air pollution in Skopje from 2008 to 2018”
(S. Petrushevski, 2018)

Структура даних

- Сезонність 24 години.
- Тренд не завжди присутній.
- Дані не завжди стаціонарні.

Рішення:

Автоматичні інструменти прогнозування



Графік сезонного розкладання(STL)

	First week	Second week
Test Statistic	-3.668150	-1.781917
p-value	0.004584	0.389475
Critical Value (1%)	-3.476927	-3.477262
Critical Value (5%)	-2.881973	-2.882118
Critical Value (10%)	-2.577665	-2.577743

Розширений тест Дікі - Фуллера

Методи

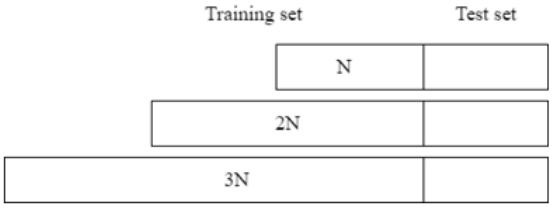
- Експоненційне згладжування(ES)
- Авторегресійна інтегрована модель ковзного середнього (ARIMA)
- Довга короткочасна пам’ять (LSTM)

Метрика оцінки точності:

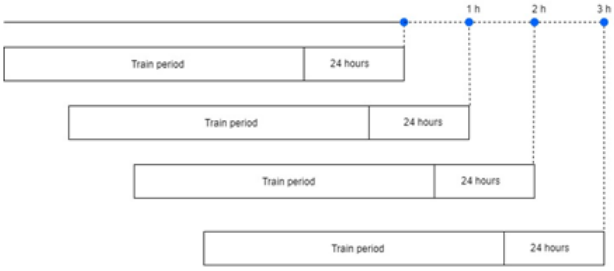
Коренева середньоквадратична похибка(RMSE)

Експеримент

- Перебудова моделі кожного разу як надходять нові дані.
- Обертовий прогноз для моделювання даних у режимі реального часу.
- Проведення експерименту на різнимх даних та усереднення результатів.



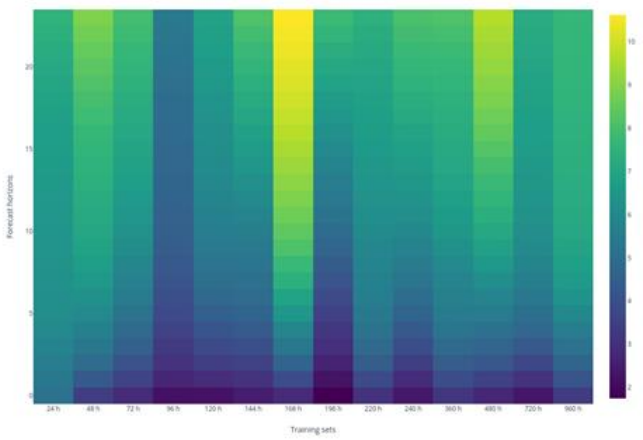
Вибір найкращого тренувального набору



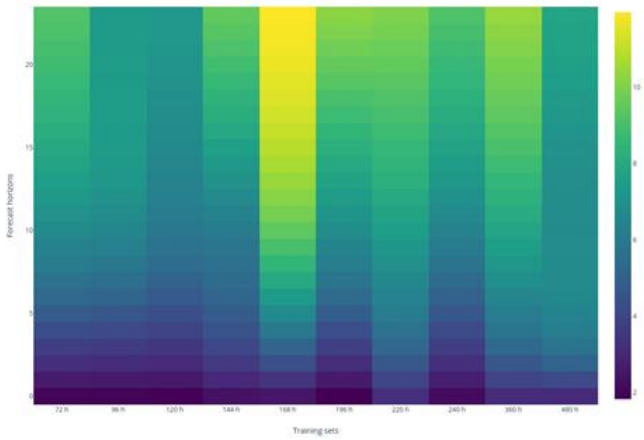
Обертовий прогноз

Результати

ES: 96 годин



ARIMA: 120 годин



Чим меньша помилка, тим темніший колір на діаграмі.

Результати

Чим більше даних отримує LSTM, тим краще прогноз (8000 годин).

Набір гіперпараметрів для найкращої моделі LSTM:

Коефіцієнт	Значення
Епохи	800
Витримка	0.1
Розмір перевіркового сету	72 год
Одноразова втрата даних	0.1
Багаторазова втрата даних	0.3
Розмір партії	12
Тип	Статичний
Коефіцієнт підрахунку одиниць	3
Розмір тренувального набору	8000 год

Результати

Експоненційне згладжування (20.5 с) та ARIMA (19.8 с) працюють ефективніше ніж LSTM (мінімально 936 с)

Висновки

У даній роботі було досліджено авторегресійні та нейромережеві методи для короткострокового передбачення забруднення повітря.

Як показали експерименти, для даної області авторегресійні методи є ефективнішими аніж нейромережеві методи.

Подальшим напрямом дослідження є аналіз інших методів прогнозування забруднення повітря та перевірка їх ефективності на практиці.

За результатами досліджень опублікована стаття на міжнародній науковій інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення»(випуск 48).

Дякую за увагу!

ДОДАТОК Б

Програмний код досліджених моделей

Б.1 Експоненційне згладжування

```
def exponential_smoothing(train, test, lambda_, trend=None,
seasonal='add', seasonal_periods=24, damped=False):
    _model = ExponentialSmoothing(train, trend=trend,
seasonal=seasonal, seasonal_periods=seasonal_periods,
                                  damped=damped)
    model_params = {
        'trend': trend,
        'seasonal': seasonal,
        'damped': damped
    }
    fit = _model.fit()
    pred = fit.forecast(len(test))
    reversed_train, reversed_test, reversed_pred =
reverse_box_cox(train, test, pred, lambda_)
    return reversed_train, reversed_test, reversed_pred,
model_params

def get_grid_search_configs():
    models = list()
    # define config lists
    trend = ['add', 'mul', None]
    damped = [True, False]
    seasonality = ['add', 'mul', None]
    for t in trend:
        for d in damped:
            for s in seasonality:
                cfg = [t, d, s]
                models.append(cfg)
    return models
```

```
def es_grid_search(train, test, lambda_):
    configs = get_grid_search_configs()
    result_list = []
    for config in configs:
        try:
            with catch_warnings():
                filterwarnings("ignore")
                result_params =
build_model_with_config(config, train, test, lambda_)
                result_list.append(result_params)
        except Exception as e:
            pass
            # result_list.append({'ERROR':config})
    best_model = select_best_model(result_list)
    return best_model
```

Б.2 ARIMA

```
def predict_auto_arima(train, test, lambda_,
                       start_p, max_p, start_q, max_q, max_d,
                       start_P, max_P, start_Q, max_Q, max_D,
                       m, information_criterion,
                       max_order=10,
                       d=None, D=None,
                       method=None, trend='c',
solver='lbfgs',
                       suppress_warnings=True,
error_action='warn', trace=True,
                       stepwise=False, seasonal=True,
n_jobs=1):
    _model = auto_arima(train, start_p=start_p, max_p=max_p,
                        d=d, max_d=max_d,
                        start_q=start_q, max_q=max_q,
```

```
                            start_P=start_P, max_P=max_P,
                            D=D, max_D=max_D,
                            start_Q=start_Q, max_Q=max_Q,
                            seasonal=seasonal, m=m,
                            max_order=max_order,
                            trace=trace,
                            error_action=error_action,
                            suppress_warnings=suppress_warnings,
                            stepwise=stepwise)
    model_params = model_params_to_json(_model)
    _model.fit(train)
    pred = _model.predict(n_periods=len(test))
    reversed_train, reversed_test, reversed_pred =
reverse_box_cox(train, test, pred, lambda_)s
    return reversed_train, reversed_test, reversed_pred,
model_params

def select_model(self):
        train_test_list = self.create_train_test_list()
        for train, test in train_test_list:

            output = {'train_start':
train.index[0].strftime('%Y-%m-%d %H-%M-%S'),
                      'train_end': train.index[-
1].strftime('%Y-%m-%d %H-%M-%S'),
                      'test_start':
test.index[0].strftime('%Y-%m-%d %H-%M-%S'),
                      'test_end': test.index[-
1].strftime('%Y-%m-%d %H-%M-%S')}
            print('BEFORE', output)
            print('BEFORE', len(train), len(test))
            model = self.build_model(train)
            output.update(self.output_model(model))
            metrics = self.predict_model(model, train, test)
            output.update(metrics)
            self.selection_result.append(output)
```

```
            print(output)
        self.print_to_file()
```

Б.3 LSTM

```
def lstm_predict(dataset, look_back):
    scaler = MinMaxScaler(feature_range=(0, 1))
    dataset = scaler.fit_transform(dataset)
    train_size = int(len(dataset) * 0.67)
    test_size = len(dataset) - train_size
    train, test = dataset[0:train_size, :],
dataset[train_size:len(dataset), :]
    # reshape into X=t and Y=t+1
    trainX, trainY = create_dataset(train, look_back)
    testX, testY = create_dataset(test, look_back)
    # reshape input to be [samples, time steps, features]
    trainX = numpy.reshape(trainX, (trainX.shape[0], 1,
trainX.shape[1]))
    testX = numpy.reshape(testX, (testX.shape[0], 1,
testX.shape[1]))
    # create and fit the LSTM network
    model = Sequential()
    model.add(LSTM(5, input_shape=(1, look_back)))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error',
optimizer='adam')
    model.fit(trainX, trainY, epochs=5, batch_size=1,
verbose=2)
    # make predictions
    trainPredict = model.predict(trainX)
    testPredict = model.predict(testX)
    # invert predictions
    trainPredict = scaler.inverse_transform(trainPredict)
    trainY = scaler.inverse_transform([trainY])
    testPredict = scaler.inverse_transform(testPredict)
```

```
    testY = scaler.inverse_transform([testY])
    # calculate root mean squared error
    trainScore = math.sqrt(mean_squared_error(trainY[0],
trainPredict[:, 0]))
    print('Train Score: %.2f RMSE' % (trainScore))
    testScore = math.sqrt(mean_squared_error(testY[0],
testPredict[:, 0]))
    print('Test Score: %.2f RMSE' % (testScore))

    measure_accuracy(testY[0], testPredict[:, 0])
    return trainPredict, testPredict

def rolling_window(df, window, shift):
    dataframes = []
    start = 0
    end = window
    while end < len(df):
        cut_df = df.iloc[start:end].copy()
        # cut_df.index = pd.to_datetime(cut_df.index)
        dataframes.append(cut_df)
        start += shift
        end += shift
    return dataframes

def show_plot(dataset, look_back, trainPredict, testPredict):
    # shift train predictions for plotting
    trainPredictPlot = numpy.empty_like(dataset)
    trainPredictPlot[:, :] = numpy.nan
    trainPredictPlot[look_back:len(trainPredict) + look_back,
:] = trainPredict
    # shift ktest predictions for plotting
    testPredictPlot = numpy.empty_like(dataset)
    testPredictPlot[:, :] = numpy.nan
    testPredictPlot[len(trainPredict) + (look_back * 2) +
1:len(dataset) - 1, :] = testPredict
    # plot baseline and predictions
```

```
plt.plot(dataset)
plt.title('LSTM')
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

ДОДАТОК В

Апробація результатів роботи

Таламанова І.С., студентка
Харківській національний університет радіоелектроніки, м.Харків, Україна
студент кафедри Програмної інженерії

ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ДЛЯ ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ ПОВІТРЯ

Забруднення повітря це одна з найголовніших ознак рівня якості життя для кожного міста, адже саме воно уражає людське здоров’я хворобами серця і легень [1]. Крім того, забруднення повітря є найбільшою причиною зміни клімату [2]. Дані ВООЗ показують, що 9 з 10 людей дихають повітрям, що містить високий рівень забруднюючих речовин.

У найближчому майбутньому ідентифікатори забруднення повітря стануть більш точними, меншими, і доступнішими для кожного жителя. Джудіт Су [3] повідомляє що прилади для вимірювання забруднення повітря можуть створити мережу раннього попередження забруднення, яка буде вміщувати в собі дуже багато інформації.

З цих причин дуже важливо оглянути методи прогнозування забруднення повітря для подальшого вибору найефективнішого методу.

Забруднення повітря є часовим рядом. Описані нижче методи є найпопулярнішими методами прогнозування часових рядів[4]. Для кожного з методів проведено аналіз їх сильних та слабких сторін.

Лінійна регресія

Лінійний підхід до моделювання взаємозв'язку між результатом (або залежною змінною) та одним або кількома наслідками (або незалежними змінними). Іншими словами, це спосіб описати зв’язок між безперервними змінними.

Плюси лінійної регресії полягають в тому ща це простий метод, нескладний у використанні. Він може обробляти різні компоненти та функції часового ряду.

Мінуси полягають в тому що метод передбачає лінійне співвідношення між залежними та незалежними змінними, яке може бути хибним та чутливим до випадаючих показників.

Експоненціальне згладжування

Техніка згладжування даних прогнозування часових рядів за допомогою функції експоненціального вікна. Існують різні типи експоненційного згладжування, які включають одно-, дво- та потрійне експоненційне згладжування.

Плюси експоненційного згладджування такі що його легко використовувати. З його допомогою можна обробити рівень, тред і сезонність змінних компонентів.

Мінусами є чутливість до випадаючих показників та дуже малий довірчий інтервал.

Інтегрована модель авторегресії та ковзного середнього (ARIMA)

ARIMA – це клас моделей прогнозування часових рядів, який розпізнає тред і сезонність. ARIMA використовує попередні спостереження для обчислення наступних термінів беручи до уваги зв'язок з відсталими термінами. Цей метод складається з двох більш простих методів: AR що розшифровується як авторегресія та MA що розшифровується як ковзне середнє.

Плюси ARIMA полягають у тому що як правило метод забезпечує точний і надійний прогноз, широкі довірчі інтервали.

Мінусом є те що метод вимагає більше спостережень, ніж інші статистичні методи.

Фільтр Калмана

Метод прогнозування систем майбутнього стану на основі попередніх станів. Він оцінює спільний розподіл ймовірності точок даних для кожного часового періоду і повертає оцінене прогнозування. Він намагається усунути помилку за статичних величин.

Плюсом є те що метод націлений на передбачення. Крім того, він не потребує стаціонарних даних, що допомагає скоротити час попередньої обробки даних. Також метод легкий у застосуванні.

Мінусом є те що фільтр Калмана передбачає лінійні залежності і найкраще працює, коли часовий ряд має розподіл Гауса.

Тета метод

Тета модель – це метод прогнозування, який застосовується до одновимірного часового ряду. В основу лежить розкладання часового ряду на дві криві, використовуючи коефіцієнти Тета, які застосовуються до другої різниці даних. Криві називаються тета-лініями і мають те саме середнє значення та нахил, що і вихідні дані. Прогноз виконується за допомогою комбінації різних тета-ліній.

Плюсами є те він легкий у використанні а також має сезонні адаптації для прогнозування сезонних даних.

Мінусом є те що прогноз може бути неточним, так як використовується велика кількістю рядів.

Лінійна динамічна система

Лінійна динамічна система представляє ще один клас методів прогнозування часових рядів. Ключова відмінність цієї моделі від моделі статичної регресії полягає в тому, що для кожного етапу коефіцієнт регресії змінюється. Лінійна динамічна система зазвичай використовується для короткострокового прогнозування та моніторингу.

Плюсом є те що її легко використовувати і розшифровувати.

Мінусом є те що для прогнозування потрібно більше часу, ніж для моделі статичної регресії.

Нейронна мережа

Нейронні мережі – це обчислювальні системи, основні принципи яких були сформовані на основі роботи людського мозку. Основна перевага полягає в тому, що НМ може «вчитися» та приймати рішення, не будучи безпосередньо запрограмованою на певну дію. НМ можна було б описати як основу багатьох алгоритмів машинного навчання для обробки даних складних структур. Існує багато типів нейронних мереж, зокрема, БП (багатошаровий персептрон), РНМ (рекурентна нейронна мережа), LSTM (довга короткочасна пам’ять). LSTM – найпопулярніша структура нейронної мережі для прогнозування часових рядів.

Плюсами є те що для нейронної мережі менше обмежень та припущень. Вона здатна обробляти складні нелінійні залежності в часовому ряді. Має високу прогнозовану силу та можливість автоматизації.

Мінусами є те що НМ має низьку інтерпретацію. Також для отримання інтервалів довіри для прогнозу потрібно багато даних.

Метод найближчого сусіда (KNN)

KNN – це простий непараметричний класифікаційний метод, який ще називають ледачим алгоритмом навчання. Метод заснований на обчисленні відстані від одного об’єкта до усіх інших об'єктів. Коефіцієнт K пояснює, скільки найближчих точок даних слід використовувати для обчислення прогнозу.

Плюсами є те що для методу найближчого сусіда легко інтерпретувати результати. Також він розраховує прогноз за короткий час.

Мінусом є те що метод вимагає багато пам’яті та зберігає всі дані, через що сповільнюється прогнозування.

Регресія опорних векторів (SVR)

Керований алгоритм машинного навчання SVR розшифровується як регресія опорних векторів. Основна ідея полягає у використанні методики під назвою хитрість ядра, яка дозволяє зробити дуже складну трансформацію даних, через що доводиться описувати дані у багатовимірному просторі. Після цього алгоритм намагається знайти найкращу криву під назвою гіперплан, яка найкращим чином розділяє дані. Прогнозування здійснюється шляхом перевірки того, наскільки близько до гіперплану знаходиться точка даних.

Плюси його такі що метод точний в багатовимірному просторі та продуктивно використовує пам’ять.

Мінусом є те що використання регресії опорних векторів може призвести до ідеалізації даних, тому прогноз не буде точним. Для великих наборів даних обчислювальна вартість дуже висока.

Оглянуті методи можуть бути використані для прогнозування забруднення повітря.

Література

1. N. Künzli, R. Kaiser, S. Medina, M. Studnicka, O. Chanel, P. Filliger, M. Herry,F. Horak Jr, V. Puybonnieux-Texier, P. Quénelet al., “Public-health impact ofoutdoor and traffic-related air pollution: a european assessment,” The Lancet, vol. 356, no. 9232, pp. 795–801, 2000.

2. J. H. Seinfeld and S. N. Pandis, Atmospheric chemistry and physics: from air pollution to climate change. John Wiley & Sons, 2016.
3. J. Su, “Portable and sensitive air pollution monitoring,” Light, science & ap-plications, vol. 7, 2018.
4. Shmueli, Galit, and Kenneth C. Lichtendahl Jr. Practical time series forecasting with r: A hands-on guide. Axelrod Schnall Publishers, 2016.