INSTITUT TEKNOLOGI SEPULUH NOPEMBER

FACULTY OF INTELLIGENT ELECTRICAL AND INFORMATICS TECHNOLOGY

Department of Information Systems

Bachelor of Computer Science in Information Systems Program

**Predictive Modelling and Analytics**

**2023**

**Final Examination**

Submitted By

**Rania Prastyka**

**5026211042**

|  |  |  |
| --- | --- | --- |
| **Submission Date** | **Mark** | **Marker** |
| **22-12-2023** | **YOUR MARK/FULL MARK** | **INITIAL MARKER** |
| **Course Convenor** | **Note** | |
| **Raras Tyasnurita** |  | |

**Declaration of Original Work**

I, your name, hereby declare that the attached individual work on Final examination is my original work. I have honored the principles of academic integrity and have upheld INSTITUT TEKNOLOGI SEPULUH NOPEMBER’s Student Code of Academic in the completion of this work.

|  |  |  |
| --- | --- | --- |
| **Date** | : | 22-12-2023 |
| **Signature** | : | Rania Prastyka |

|  |  |  |
| --- | --- | --- |
| **Point** | Your Answer | Mark |
| 1 | **Defining data pattern:**  Evaluasi terhadap kualitas udara merupakan cara penting untuk memantau dan mengendalikan polusi udara. Karakteristik pasokan udara mempengaruhi kesesuaiannya untuk penggunaan tertentu. Beberapa polutan udara dapat mengganggu kesehatan, merusak lingkungan, dan menyebabkan kerugian.  Dataset yang saya gunakan merupakan data untuk kondisi kualitas udara di Negara Italia pada Tahun 2004-2005. Berikut informasi lengkap untuk dataset Air Quality yang saya gunakan:  Dataset ini dapat diakses di: <https://archive.ics.uci.edu/dataset/360/air+quality>  Berikut gambaran untuk isi dataset Air Quality yang saya gunakan  Informasi mengenai tipe data:  Pada Kolom Date sudah memiliki tipe data datetime, namun untuk kolom waktu belum. Sehingga, saya akan melakukan perubahan tipe data untuk kolom ‘Time’ menjadi tipe datetime dan menggabungkannya dengan kolom ‘Date’ menjadi satu kolom yang Bernama ‘Datetime’ menggunakan query sebagai berikut:  df = pd.DataFrame(data)  # Mengonversi kolom Time menjadi tipe data datetime  df['Time'] = pd.to\_datetime(df['Time'], format='%H:%M:%S').dt.time  # Menggabungkan kolom Date dan Time menjadi satu kolom Datetime  df['Datetime'] = pd.to\_datetime(df['Date'].astype(str) + ' ' + df['Time'].astype(str))  # Pindahkan kolom Datetime ke posisi pertama  cols = list(df.columns)  cols.insert(0, cols.pop(cols.index('Datetime')))  df = df[cols]  # Menghapus kolom lama Date dan Time  df = df.drop(['Date', 'Time'], axis=1)  Sehingga dihasilkan struktur data baru sebagai berikut:    Sekarang, saya akan mengubah RangeIndex pada data tersebut dengan DatetimeIndex, hal tersebut dilakukan untuk mempermudah dalam analisis deret waktu khusunya untuk melakukan prediksi  df.set\_index('Datetime', inplace=True)  Hasil:    Secara keseluruhan, data tersebut mempunyai 13 kolom dan tidak ada kolom yang null. Untuk datatype terdiri dari tipe float sebanyak 12 kolom dan tipe integer sebanyak 1 kolom.  Untuk informasi tambahan berikut deskripsi data untuk dataset tersebut, deskripsi ini merupakan informasi terkait jumlah, rata-rata, nilai maxmin, dll. Ringkasan ini memberikan informasi statistik penting yang dapat membantu dalam memahami distribusi data dalam setiap kolom numerik.    Untuk lebih mudah memahami berikut definisi untuk setiap kolom yang ada:   * Date: Tanggal pengukuran. * Time: Waktu pengukuran. * CO(GT): Konsentrasi karbon monoksida dalam Gram per meter kubik (g/m³). * PT08.S1(CO): Nilai sensor pertama terkait karbon monoksida. * NMHC(GT): Konsentrasi hidrokarbon non-metana dalam Gram per meter kubik (g/m³). * C6H6(GT): Konsentrasi benzene dalam Gram per meter kubik (g/m³). * PT08.S2(NMHC): Nilai sensor kedua terkait hidrokarbon non-metana. * NOx(GT): Konsentrasi nitrogen oksida dalam Gram per meter kubik (g/m³). * PT08.S3(NOx): Nilai sensor ketiga terkait nitrogen oksida. * NO2(GT): Konsentrasi nitrogen dioksida dalam Gram per meter kubik (g/m³). * PT08.S4(NO2): Nilai sensor keempat terkait nitrogen dioksida. * PT08.S5(O3): Nilai sensor kelima terkait ozon. * T: Temperatur dalam derajat Celsius. * RH: Kelembaban relatif dalam persen. * AH: Tekanan udara dalam bukan kelembaban (sebuah parameter yang berkaitan dengan kelembaban).   Setelah mengkaji lengkap untuk informasi data, berikut pola persebaran data berdasarkan waktu untuk setiap bulannya.    Dari pola perserbaran data untuk setiap kolom dan berdasarkan bulan, dapat dilihat untuk variabel konsentrasi hidrokarbon non-metana dalam Gram per meter kubik (g/m³) atau kolom NMHC(GT) cenderung stabil ketika memasuki bulan juni 2004 sampai bulan mei 2005, karena tidak ada tren naik atau turun. Untuk variabel lain cenderung mempunyai pola musiman dimana pola menunjukkan fluktuasi berulang dimana terjadi kenaikan dan penurunan dalam waktu tertentu  Berikut box plot untuk pola perserbaran data bulanan pada data tersebut    Melihat korelasi antar variabel    Berikut hasil dari analisis matriks korelasi tersebut:   * Kolom 'CO(GT)' memiliki Korelasi Tertinggi dengan 'NO2(GT)' sebesar 0.67 * Kolom 'PT08.S1(CO)' memiliki Korelasi Tertinggi dengan 'PT08.S2(NMHC)' sebesar 0.93 * Kolom 'NMHC(GT)' memiliki Korelasi Tertinggi dengan 'PT08.S1(CO)' sebesar 0.17 * Kolom 'C6H6(GT)' memiliki Korelasi Tertinggi dengan 'AH' sebesar 0.98 * Kolom 'PT08.S2(NMHC)' memiliki Korelasi Tertinggi dengan 'PT08.S1(CO)' sebesar 0.93 * Kolom 'NOx(GT)' memiliki Korelasi Tertinggi dengan 'NO2(GT)' sebesar 0.82 * Kolom 'PT08.S3(NOx)' memiliki Korelasi Tertinggi dengan 'AH' sebesar 0.62 * Kolom 'NO2(GT)' memiliki Korelasi Tertinggi dengan 'NOx(GT)' sebesar 0.82 * Kolom 'PT08.S4(NO2)' memiliki Korelasi Tertinggi dengan 'PT08.S2(NMHC)' sebesar 0.87 * Kolom 'PT08.S5(O3)' memiliki Korelasi Tertinggi dengan 'PT08.S2(NMHC)' sebesar 0.91 * Kolom 'T' memiliki Korelasi Tertinggi dengan 'AH' sebesar 0.98 * Kolom 'RH' memiliki Korelasi Tertinggi dengan 'AH' sebesar 0.94 * Kolom 'AH' memiliki Korelasi Tertinggi dengan 'C6H6(GT)' sebesar 0.98   Setelah mendapat korelasi tersebut, selanjutnya yaitu melakukan cleaning data dengan menghapus kolom yang tidak perlukan. Namun, untuk korelasi antar variabel tersebut semua data dibutuhkan untuk analisis  Selanjutnya yaitu menangani untuk nilai-nilai yang hilang/kosong.   * Mendeteksi nilai yang null menggunakan plot     Dari hasil tersebut tidak ada data yang null(kosong), sehingga dapat langsung dilakukan proses prediksi data |  |
| 2 | **LTSM**  Applying Forecasting Method:   1. Menyiapkan data input training model   # Split a multivariate sequence into samples  def split\_sequences(sequences, n\_steps\_in, n\_steps\_out):    X, y = list(), list()    for i in range(len(sequences)):      # find the end of this pattern      end\_ix = i + n\_steps\_in      out\_end\_ix = end\_ix + n\_steps\_out      # check if we are beyond the dataset      if out\_end\_ix > len(sequences):        break      # gather input and output parts of the pattern      seq\_x, seq\_y = sequences[i:end\_ix, :], sequences[end\_ix:out\_end\_ix, :]      X.append(seq\_x)      y.append(seq\_y)    return array(X), array(y)  # split dataset to train and test  total\_dataset = len(df)  df\_train = df[:int(total\_dataset\*0.76)]  df\_test = df[int(total\_dataset\*0.76):total\_dataset]  # scaled  scaler = MinMaxScaler()  df\_train\_scaled = scaler.fit\_transform(df\_train)  df\_test\_scaled = scaler.fit\_transform(df\_test)  print('Data for train:', df\_train.shape)  print('Total day for train:', df\_train.shape[0]/24)  print('\nData for test:', df\_test.shape)  print('Total day for test:', df\_test.shape[0]/24)  Data for train: (7111, 13)  Total day for train: 296.2916666666667  Data for test: (2246, 13)  Total day for test: 93.58333333333333  **Untuk data train berjumlah 296 hari, sedangkan data test berjumlah 93 hari.**  # n\_steps\_in history data for predict n\_steps\_out forecast data  n\_steps\_in, n\_steps\_out = 6, 3  # Convert into input/output  x\_train, y\_train = split\_sequences(df\_train\_scaled, n\_steps\_in, n\_steps\_out)  x\_test, y\_test = split\_sequences(df\_test\_scaled, n\_steps\_in, n\_steps\_out)  # The dataset knows the number of features, e.g. 2  n\_features = x\_train.shape[2]  **df\_train\_scaled** dan **df\_test\_scaled** adalah DataFrame yang berisi data deret waktu yang telah dinormalisasi (misalnya, menggunakan Min-Max Scaling), dan **split\_sequences** adalah fungsi yang membagi data menjadi langkah-langkah waktu yang sesuai, **maka x\_train, y\_train, x\_test,** dan **y\_test** akan digunakan sebagai input dan output dalam melatih serta menguji model LSTM atau model deret waktu serupa. n\_step\_in dan n\_step\_out dalam syntax tersebut yaitu kita menggunakan 6 data terdahulu untuk 3 data prediksi  Setelah itu, berikut ditampilkan struktur data input dan output yang dihasilkan setelah mempersiapkan data menggunakan split\_sequences.    **Seperti yang telah di atur di n step, jadi data input mempunyai 6 data untuk melatih model dan data outputnya ada 3 untuk validasi dari model itu sendiri**  print('Total features:', n\_features)  print('Total train data:', x\_train.shape)  print('Total validation train data:', y\_train.shape)  print('Total test data:', x\_test.shape)  print('Total validation test data:', y\_test.shape)     1. Melakukukan train  * Mendefinisikan arsitektur model LSTM   Mengimplementasikan pendekatan encoder-decoder menggunakan dua lapisan LSTM: satu sebagai encoder untuk mengekstraksi representasi tersembunyi dari data input, dan satu sebagai decoder untuk menghasilkan prediksi langkah waktu ke depan. RepeatVector digunakan untuk mengulang representasi tersembunyi sebagai input ke dalam lapisan decoder. Lapisan TimeDistributed memastikan bahwa setiap langkah waktu memiliki layer Dense terpisah untuk prediksi fitur.      Ringkasan ini memberikan informasi singkat tentang nama model, struktur dan parameter setiap lapisan, serta total parameter yang digunakan dalam model.  Selanjutnya, melakukan fit model agar dapat melakukan prediksi yang akurat pada data. Proses ini mengoptimalkan parameter-model berdasarkan data pelatihan dan mengukur sejauh mana model mampu generalisasi pada data yang belum pernah dilihat sebelumnya (data validasi). Dengan demikian, model dapat menjadi lebih baik dalam melakukan prediksi  Menggunakan Query sebagai berikut:  # fit model  epoch = 50  model.compile(optimizer='adam', loss='mse',                metrics=[tf.metrics.MeanAbsoluteError()])  history = model.fit(x\_train, y\_train, epochs=epoch, validation\_data=(x\_test, y\_test))  dari 50, berikut hasil yang ditampilkan (10/50)  **Loss:** Menunjukkan sejauh mana nilai prediksi model berbeda dari nilai sebenarnya pada data pelatihan dan data validasi. Nilai loss pada data tersebut menunjukkan nilai kecil  **MAE (Mean Absolute Error):** Menunjukkan rata-rata dari nilai absolut dari selisih antara prediksi dan nilai sebenarnya pada data pelatihan dan data validasi. Semakin kecil MAE, semakin baik model memprediksi data. Pada hasil yang ada rata-rata pada nilai 0.05   1. Melakukan Tes Model   Setelah pelatihan, saya akan melakukan tes model dengan membuat variabel baru.    **predict\_generator** digunakan untuk memprediksi output berdasarkan generator atau input generator. Dalam hal ini, generator x\_test digunakan untuk membuat prediksi.  (2238, 3, 13) pertama:   * bentuk dari array hasil prediksi (predictions). * Terdapat 2238 sampel data uji. * Setiap sampel memiliki 3 langkah waktu (3 time steps). * Untuk setiap langkah waktu, terdapat 13 nilai prediksi (13 features).   (2238, 3, 13) kedua:   * Bentuk dari array data output sebenarnya pada data uji (y\_test). * Juga, terdapat 2238 sampel data uji. * Setiap sampel memiliki 3 langkah waktu. * Untuk setiap langkah waktu, terdapat 13 nilai yang merepresentasikan output sebenarnya.   Melakukan reshape agar bisa dijadikan data frame. Lalu juga melakukan reschale    Menampilkan hasil prediksi, namun untuk hasil dibawah kolom belum terdefinisi secara jelas (berupa angka)    Begitu juga dengan data real, kolom masih berupa angka      Menggabungkan data prediksi dengan data real, dengan membuat dataframe baru  didapatkan hasil sebagai berikut:  Memvisualisasikan data frame atau data\_final untuk melihat seberapa akurat hasil prediksi:                             1. Melihat metrics eror dari hasil prediksi   **RMSE (Root Mean Squared Error):**   * RMSE mengukur seberapa baik model dapat memprediksi nilai sebenarnya. * Semakin rendah RMSE, semakin baik model dapat memprediksi nilai sebenarnya. * Misalnya, pada variabel "CO(GT)", RMSE sekitar 35.62 menunjukkan bahwa model memiliki deviasi rata-rata sekitar 35.62 dari nilai sebenarnya.   **MSE (Mean Squared Error):**   * MSE adalah rata-rata dari kuadrat selisih antara nilai prediksi dan nilai sebenarnya. * Nilai MSE yang lebih rendah menunjukkan bahwa model memiliki deviasi yang lebih kecil dari nilai sebenarnya.   **MAE (Mean Absolute Error):**   * MAE adalah rata-rata dari nilai absolut selisih antara nilai prediksi dan nilai sebenarnya. * MAE memberikan gambaran tentang besarnya deviasi absolut antara nilai prediksi dan nilai sebenarnya.   **MAPE (Mean Absolute Percentage Error):**   * MAPE mengukur persentase rata-rata kesalahan absolut relatif terhadap nilai sebenarnya. * MAPE yang rendah menunjukkan bahwa model memiliki kesalahan relatif yang kecil.   **Interpretasi Hasil:**  Dalam interpretasi hasil ini, dapat dilihat kinerja model pada setiap variabel target. Misalnya, model memiliki kinerja yang baik dalam memprediksi konsentrasi "NMHC(GT)" (MAPE sangat rendah), tetapi memiliki kesalahan yang lebih besar dalam memprediksi variabel "PT08.S5(O3)" (MAPE lebih tinggi).   1. feget   Hasil forecast atau prediksi yang telah saya lakukan diatas merupakan forecast untuk keseluruhan variabel, lalu Bagaimana jika prediksi dilakukan untuk salah satu variabel dependent yang dipengaruhi oleh variabel lain(independent)?. Berikut Langkah-langkah dan hasil forecasting yang saya gunakan menggunakan beberapa metode seperti Linear Regression, Logistic Regression, Decision Tree Regression, Random Forest Regression,  [Link Forecast LTSM](https://colab.research.google.com/drive/1xW62fEcYbBJITxW52RdFwlGwOYmaBYMU?usp=sharing)  **LINEAR REGRESSION**   1. melakukan pembersihan data      1. Membuat kolom baru yang menyimpan informasi bulan (MONTH) dari kolom tanggal (DATE) pada DataFrame df1. 2. membuat kolom baru bernama 'HOUR' dalam DataFrame df1, yang berisi nilai jam yang diekstrak dari kolom 'TIME'.      1. Data setelah ada kolom baru ‘MONTH’ dan ‘HOUR’ 2. Melihat korelasi antar variabel      1. Variabel dependent yang digunakan yaitu variabel RH, disini kita akan memahami linearitas antara kelembapan relaltif (RH) dan variabel lainnya                                  1. menghapus kolom 'RH' dari DataFrame tersebut dan membuat Series Y yang berisi kolom 'RH' dari DataFrame df1.      1. Menampilkan kolom Y yang berisi RH      1. membagi dataset menjadi dua bagian: satu untuk pelatihan model (training) dan satu untuk pengujian model (testing).   X\_train, X\_test, y\_train, y\_test=train\_test\_split(Xsd,Y,test\_size=0.4, random\_state=3)  X\_train: Fitur untuk pelatihan model.  X\_test: Fitur untuk pengujian model.  y\_train: Label untuk pelatihan model.  y\_test: Label untuk pengujian model.   1. mencetak atau menampilkan ukuran (shape) dari data pelatihan (training data) dan data uji (test data)  * Data Pelatihan (Training Data):   Jumlah sampel (baris): 5614  Jumlah fitur (kolom): 14   * Data Uji (Test Data):   Jumlah sampel (baris): 3743  Jumlah fitur (kolom): 14   1. Mengeskekusi model   Proses ini menggambarkan penggunaan model regresi linear untuk membuat prediksi berdasarkan variabel-variabel fitur yang diberikan. Hasil prediksi (prediction) dapat dibandingkan dengan nilai sebenarnya (y\_test) untuk mengevaluasi performa mode.     1. Mencetak koefisien dari model regresi linear untuk masing-masing variabel independen.      1. menghitung dan mencetak Root Mean Squared Error (RMSE)   RMSE Linear Regresion: 6.971372753947364  **LOGISTIC REGRESSION**  Proses train dan test data hamper sama dengan linear regression, hanya saja model disiapkan dengan syntax logistic regression  Berikut Hasil prediksi untuk Logistic Regression  RMSE= 1874.3410525361783  **DECISION TREE REGRESION**    RMSE= 1.380210329925694  **RANDOM FOREST REGRESSION**  RMSE= 0.696116687199154  **SUPPORT VECTOR MACHINE**  RMSE= 24.502591458122172  **Link Hasil Forecast liner regression, etc.**  <https://colab.research.google.com/drive/19y13UJoRbX-tV06Qe1Wg_GWSHWNA8HST?usp=sharing> |  |
| 3 | **Kesimpulan:**  Setelah melakukan forecasting dengan menggunakan beberapa metode, di dapat kesimpulan sebagai berikut:   1. Forecasting atau prediksi menggunakan metode LTSM dengan semua varibel dilakukan prediksi memberikan hasil dengan keakuratan yang berbeda-beda. Hal tersebut dilihat dari nilai RMSE yang dihasilkan 2. Setelah dilakukan prediksi untuk variabel ‘RH’ dengan menggunakan beberapa metode, ternyata model yang paling baik memberikan hasil predisiki adalah random forest. Hal tersebut karena random forest memiliki nilai RMSE yang paling rendah. Artinya, tingkat keakuratan model ini juga semakin tinggi.   **Rekomendasi:**  Untuk dataset AirQuality ini direkomendasikan untuk melakukan prediksi kualitas udara negara italia menggunakan metode **random forest** |  |
| TOTAL MARK | |  |