ANALITIK BIG DATA DENGAN PYSPARK

Penerapan Algoritma Logistic Regression pada Dataset weatherAUS untuk Prediksi Hujan Di Australia

Kelompok 6

1. **NURUL NAJWA SABILLA (312110451)**
2. **LAELA NUR ROHMAH (312110425)**
3. **ALVINA DAMAYANTI (312110125)**
4. **SARA KHUSNUL MUMTAZAH (312110319)**

Kelas TI.21.A.3

Pendahuluan

01.
Gambaran singkat tentang Big Data Analytics dan PySpark

Big Data Analytics adalah proses yang digunakan untuk mengambil pola tersembunyi, korelasi yang tidak diketahui, tren pasar, dan preferensi pelanggan.

PySpark adalah library open-source Python yang menyediakan antarmuka untuk Apache Spark, sebuah kerangka kerja pengolahan data berdistribusi.

02.
Mengapa kita memilih dataset weatherAUS untuk analisis?

Karena kasusnya dataset weatherAUS yang digunakan untuk memprediksi apakah akan terjadi hujan besok berdasarkan data cuaca pada hari sebelumnya.
Metode machine learning yang digunakan adalah Logistic Regression karena metode ini sangat popular digunakan.

03.

Tujuan Presentasi

Dapat menghasilkan model yang memiliki akurasi yang baik dalam memprediksi hujan pada esok hari di wilayah Australia.

PENERAPAN ALGORITMA LOGISTIC REGRESSION

1 PENGENALAN ALGORITMA LOGISTIC REGRESSION

Logistic Regression adalah algoritme pembelajaran mesin terawasi (pembelajaran mesin di mana model atau algoritma diajarkan dari data yang telah diketahui labelnya) yang terutama digunakan untuk tugas klasifikasi yang tujuannya adalah memprediksi kemungkinan suatu instance termasuk dalam kelas tertentu atau tidak.

2 CARA LOGISTIC REGRESSION DIGUNAKAN DALAM PEMODELAN PREDIKSI APAKAH HUJAN BESOK (RAINTOMMOROW) PADA DATASET WEATHERAUS

1. Melakukan skema awal untuk dataset

```
In [4]: df.dtypes
Out[4]: [('Date', 'date'),
         ('Location', 'string'),
         ('MinTemp', 'string'),
         ('MaxTemp', 'string'),
         ('Rainfall', 'string'),
         ('Evaporation', 'string'),
         ('Sunshine', 'string'),
         ('WindGustDir', 'string'),
         ('WindGustSpeed', 'string'),
         ('WindDir9am', 'string'),
         ('WindDir3pm', 'string'),
         ('WindSpeed9am', 'string'),
         ('WindSpeed3pm', 'string'),
         ('Humidity9am', 'string'),
         ('Humidity3pm', 'string'),
         ('Pressure9am', 'string'),
         ('Pressure3pm', 'string'),
         ('Cloud9am', 'string'),
         ('Cloud3pm', 'string'),
         ('Temp9am', 'string'),
         ('Temp3pm', 'string'),
         ('RainToday', 'string'),
         ('RainTomorrow', 'int')]
```

2. Merapikan dataset sesuai skema awal yang dibuat dengan cara meng-convert type data, serta hasilnya:

```
In [74]: df.printSchema()
root
 |-- Date: string (nullable = true)
 |-- Location: string (nullable = true)
 |-- MinTemp: float (nullable = true)
 |-- MaxTemp: float (nullable = true)
 |-- Rainfall: float (nullable = true)
 |-- Evaporation: float (nullable = true)
 |-- Sunshine: float (nullable = true)
 |-- WindGustDir: string (nullable = true)
 |-- WindGustSpeed: float (nullable = true)
 |-- WindDir9am: string (nullable = true)
 |-- WindDir3pm: string (nullable = true)
 |-- WindSpeed9am: float (nullable = true)
 |-- WindSpeed3pm: float (nullable = true)
 |-- Humidity9am: float (nullable = true)
 |-- Humidity3pm: float (nullable = true)
 |-- Pressure9am: float (nullable = true)
 |-- Pressure3pm: float (nullable = true)
 |-- Cloud9am: float (nullable = true)
 |-- Cloud3pm: float (nullable = true)
 |-- Temp9am: float (nullable = true)
 |-- Temp3pm: float (nullable = true)
 |-- RainToday: string (nullable = true)
 |-- RainTomorrow: integer (nullable = false)
```

3.Clening data sebelum dilakukan tindakan selanjutnya, yaitu mengisi missing value

Cek Missing Value

```
In [16]: missing_value_sum = df.select([sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in df.columns])
         missing_value_sum.show()
```

```
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|Date|Location|MinTemp|MaxTemp|Rainfall|Evaporation|Sunshine|WindGustDir|WindGustSpeed|WindDir9am|WindDir3pm|WindSpeed9am|WindSpeed3pm|Humidity9am|Humidity3pm|Pressure9am|Pressure3pm|Cloud9am|Cloud3pm|Temp9am|Temp3pm|RainToday|RainTomorrow|
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|   0|       0|   1485|   1261|    3261|      62790|   69835|          0|        10263|         0|         0|       1767|       3062|       2654|       4507|      15065|      15028|   55888|   59358|   1767|   3609|        0|           0|
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
```

Cleaning dilakukan dengan imputasi mean, median, modus, sehingga data bebas dari kekosongan data

```
In [24]: missing_value_count = df.select([sum(when(col(c).isNull(), 1).otherwise(0)).alias(c) for c in df.columns])
         missing_value_count.show()
```

```
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|Date|Location|MinTemp|MaxTemp|Rainfall|Evaporation|Sunshine|WindGustDir|WindGustSpeed|WindDir9am|WindDir3pm|WindSpeed9am|WindSpeed3pm|Humidity9am|Humidity3pm|Pressure9am|Pressure3pm|Cloud9am|Cloud3pm|Temp9am|Temp3pm|RainToday|RainTomorrow|
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|   0|       0|      0|      0|       0|          0|       0|          0|            0|         0|         0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|          0|
+----+--------+-------+-------+--------+-----------+--------+-----------+-------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
```

4. Pada tahap cleaning ini variabel berganti dari “df” jadi “numerical”. Variabel numerical ini akan dijadikan features dalam model dan target itu yaitu “RainTommorow”

```
In [66]: assembler = VectorAssembler(inputCols=numerical, outputCol='features')
         output = assembler.transform(df)
         output = output.select('features', 'RainTomorrow')
         output = output.withColumnRenamed('RainTomorrow', 'label')

         output.show(10)
         # numerical
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[13.3999996185302...|    0|
|[7.40000009536743...|    0|
|[12.8999996185302...|    0|
|[9.19999980926513...|    0|
|[17.5,32.29999923...|    0|
|[14.6000003814697...|    0|
|[14.3000001907348...|    0|
|[7.69999980926513...|    0|
|[9.69999980926513...|    1|
|[13.1000003814697...|    0|
+--------------------+-----+
only showing top 10 rows
```

3 LANGKAH-LANGKAH IMPLEMENTASI LOGISTIC REGRESSION DENGAN PYSPARK

1. Bagi DataFrame menjadi train_data (80%) dan test_data (20%)

```
In [77]: train_data, test_data = output.randomSplit([0.8, 0.2])

         # Cetak dimensi dari train_data dan test_data
         print('Train Shape: ({}, {})'.format(train_data.count(), len(train_data.columns)))
         print('Test Shape: ({}, {})'.format(test_data.count(), len(test_data.columns)))
```

```
Train Shape: (116365, 2)
Test Shape: (29095, 2)
```

2. Melakukan modelling Logistic Regression

Modelling

Lakukan percobaan model dengan LogisticRegression

```
In [80]: my_log_reg_model = LogisticRegression()

         fitted_logreg = my_log_reg_model.fit(train_data)
         predictions_and_labels = fitted_logreg.evaluate(test_data)
         # print(predictions_and_labels.accuracy)
         predictions_and_labels.predictions.show(10)
```

```
+--------------------+-----+--------------------+--------------------+----------+
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|[-5.3000001907348...|    0|[19.2844569365475...|[0.99999999578432...|       0.0|
|[-5.1999998092651...|    0|[19.0894712202619...|[0.99999999487672...|       0.0|
|[-5.0,8.899999618...|    0|[19.2332367547975...|[0.99999999556277...|       0.0|
|[-5.0,15.19999980...|    0|[19.7188122669545...|[0.99999999726958...|       0.0|
|[-4.9000000953674...|    0|[19.4861156871368...|[0.99999999655422...|       0.0|
|[-4.6999998092651...|    0|[19.1185296676488...|[0.99999999502345...|       0.0|
|[-4.5,12.80000019...|    0|[19.7363381712157...|[0.99999999731702...|       0.0|
|[-4.5,16.10000038...|    1|[-17.469835237188...|[2.58789675184994...|       1.0|
|[-4.4000000953674...|    0|[19.3068206560686...|[0.99999999587756...|       0.0|
|[-4.1999998092651...|    0|[19.6256448590304...|[0.99999999700297...|       0.0|
+--------------------+-----+--------------------+--------------------+----------+
only showing top 10 rows
```

3. Lalu melakukan evaluate apakah dataset ini cocok dengan model yang dipergunakan, hasilnya adalah 0.9999998149655362. Hasil ini mendekati 1 bahwa model klasifikasi biner memiliki kinerja yang sagat baik dalam membedakan antar kelas positf dan negative (terjadi hujan besok dan tidak terjadi hujan besok)

```
In [69]: my_eval = BinaryClassificationEvaluator()
         my_final_roc = my_eval.evaluate(predictions_and_labels.predictions)
         my_final_roc
Out[69]: 0.9999998149655362
```

HASIL ANALISIS

1 RINGKASAN HASIL ANALISIS LOGISTIC REGRESSION PADA DATASET WEATHERAUS

hasil dari evaluasi sebelumnya metrik ROC (Receiver Operating Characteristic) yang dihitung oleh BinaryClassificationEvaluator. Nilai ROC ini mendekati 1, yang menunjukkan bahwa model klasifikasi biner Anda memiliki kinerja yang sangat baik dalam membedakan antara kelas positif dan kelas negatif. Dalam konteks evaluasi metrik ROC, nilai AUC-ROC mendekati 1 (atau 100%) menunjukkan bahwa model klasifikasi memiliki kinerja yang sangat baik dalam membedakan antara dua kelas yang berbeda.

2 VISUALISASI HASIL, SEPERTI GRAFIK HUBUNGAN VARIABEL PREDIKTOR DAN VARIABEL TARGET

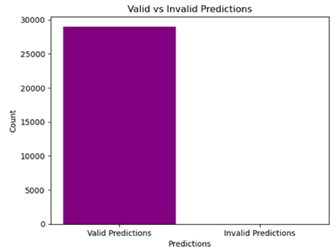
Visualization

```
In [62]: valid_predictions_count = predictions_and_labels.predictions[col("prediction") == col("label")].count()
         invalid_predictions_count =  predictions_and_labels.predictions[col("prediction") != col("label")].count()
         accuracy_predictions_count = valid_predictions_count / predictions_and_labels.predictions.count() * 100

         print("akurasi prediksi yang bener dari label dan prediction: " + str(valid_predictions_count))
         print("akurasi prediksi yang salah dari label dan prediction: " + str(invalid_predictions_count))
         print("akurasi untuk prediksi yang benar: "+ str(accuracy_predictions_count))

         akurasi prediksi yang bener dari label dan prediction: 28847
         akurasi prediksi yang salah dari label dan prediction: 0
         akurasi untuk prediksi yang benar: 100.0
```

Hasil output menunjukkan bahwa dari seluruh prediksi yang telah dilakukan, tidak ada yang salah (0 prediksi yang salah). Seluruh prediksi (28847 prediksi) cocok atau sesuai dengan nilai label yang seharusnya. Dalam istilah akurasi, ini berarti bahwa model atau algoritma yang digunakan untuk membuat prediksi memiliki akurasi sebesar 100%.

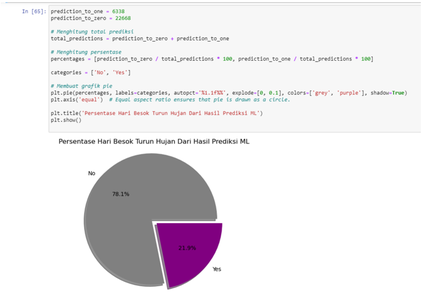
Berikut ini visualisasinya

```
In [63]: valid_predictions_count = 29006
         invalid_predictions_count = 0

         # Menyiapkan data
         categories = ['Valid Predictions', 'Invalid Predictions']
         counts = [valid_predictions_count, invalid_predictions_count]

         # Membuat grafik batang
         plt.bar(categories, counts, color=['purple', 'grey'])
         plt.xlabel('Predictions')
         plt.ylabel('Count')
         plt.title('Valid vs Invalid Predictions')
         plt.show()
```



1. Label

2. Prediksi



3 INTERPRETASI HASIL DAN TEMUAN PENTING

Dengan kata lain, model ini berhasil memprediksi semua data dengan benar sesuai dengan label yang ada. Ini adalah hasil yang sangat baik dan menunjukkan bahwa model tersebut sangat andal dalam melakukan prediksi pada dataset yang digunakan. Akurasi sebesar 100% adalah tujuan yang sangat diinginkan dalam banyak kasus, tetapi juga perlu memeriksa apakah ini adalah hasil yang sesuai dengan ekspektasi atau apakah mungkin ada masalah seperti overfitting.

Hasil dan Kesimpulan

1 TEMUAN PENTING DAN INSIGHT (WAWASAN) YANG DIPEROLEH

- Mengetahui klasifikasi data pada logistic regression berguna memprediksi label data atau target
- Lalu, mengetahu probabilitas prediksi yang menjadi output. Hal ini memungkinkan kita untuk mengukur tingkat keyakinan atau kepastian dalam prediksi.
- Serta logistic regression itu membutuhkan metrik evaluasi untuk mengukur kinerja model salah satunya dengan metric ROC.

2 KESIMPULAN DAN SARAN

Berdasarkan hasil dataset wetherAUS ini, kemungkinan terjadi hujan pada hari esok hari di Australia diprediksi dengan logistic regression mendekati 1, yang menunjukkan bahwa model sangat baik. Lalu, melihat akurasi antara hubungan variabel prediktor/prediksi dan variabel target menjadi metrik evaluasi kinerja model untuk mengukur sejauh mana model dapat membedakan actual label dengan prediksi hasil ML adalah 100%. Namun, hal ini bisa menimbulkan masalah yaitu overfitting (algoritma ML terlalu spesifik terhadap data). Saran mengenai hal itu dapat melakukan penyederhanaan model dengan mengganti algoritma untuk mencari model yang terbaik.