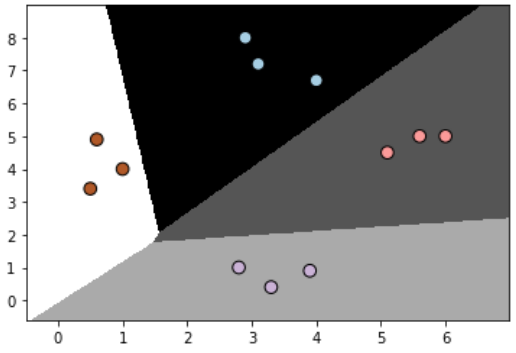
Nama : Ivan Munandar Purnama
NIM : 3332190035
Kelas : Kecerdasan Buatan B

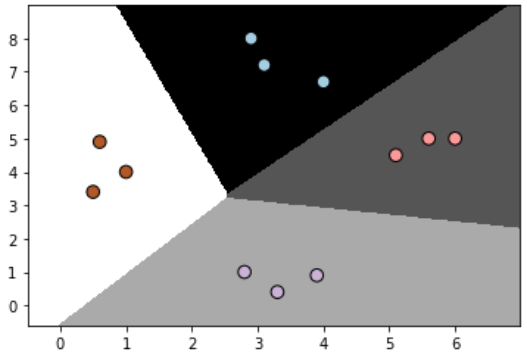
Kecerdasan Buatan

1. Analisa logistic_regression.py

Logistic regression adalah algoritma supervised learning yang digunakan untuk melakukan pengklasifikasian. Algoritma analisis logistic regression merupakan algoritma analisis prediktif yang memanfaatkan konsep probabilitas dan mengklasifikasikan data baru menggunakan dataset kontinyu dan diskrit. Pada gambar hasil penglasifikasian menggunakan algoritma regresi logistik didapakan bahwa pengklasifikasian akan semakin baik jika nilai C semakin besar.



Gambar 1 C=1



Gambar 2 C=100

Pada algoritma logistic regresi yang diberikan, kita perlu untuk melakukan import library yang dibutuhkan terlebih dahulu seperti library numpy yang digunakan untuk operasi vektor dan matriks, library skylern yang digunakan untuk melakukan processing, library matplotlib yang digunakan untuk menvisualisasikan data. Adapun library utilites yang digunakan sebagai pemvisualisasikan klasifikasi data. Utilities ini merupakan program python yang digunakan sebagai library.

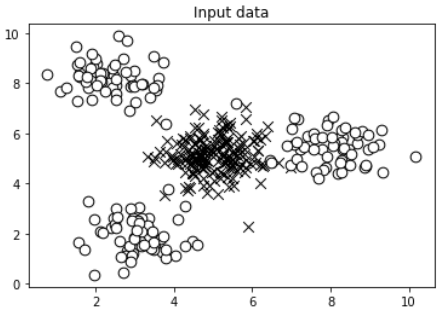
Setelah library di importkan kedalam cell atau workspace kita perlu untuk memasukan dataset yang kita perlu. Pada program yang diberikan, dataset yang digunakan berupa array yang diinputkan ke dalam variabel x dan y. Selanjutnya daset ini dilakukan training mengunakan pemodelan "`linear_model.LogisticRegression`". Pada saat proses train kita

melakukan training dengan jumlah C berbeda untuk melihat pengaruhnya pada hasil training. C pertama yaitu 1, sedangkan C kedua yaitu 100. Setelah melalui proses training kemudian data hasil prediksi di plotkan menggunakan syntax "*visualize_classifier(classifier, X, y)*". Hasil dari training algoritma logic regression dapat dilihat pada gambar 1 dan 2. Berdasarkan gambar dapat dilihat bahwa hasil training dengan C=100 lebih baik dari pada C=1. Dikatakan baik karena domain klasifikasinya lebih rapi dan tersusun.

2. Analisa decission_trees.py

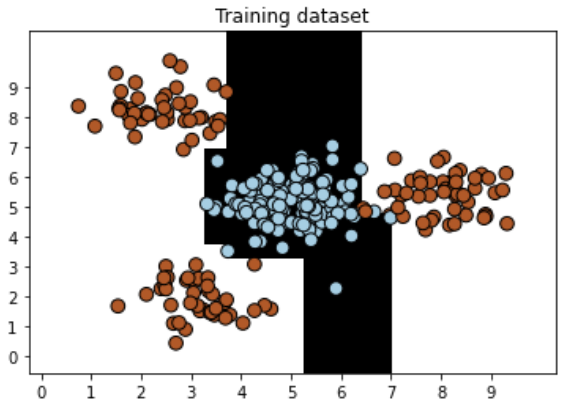
Algoritma Decision tree akan memprediksi data dengan cara mengklasifikasikannya kedalam bentuk struktur pohon. Konsep dari decision tree yaitu dengan memecah data yang dinyatakan dalam bentuk atribut dan record menjadi himpunan bagian yang lebih kecil. Tujuan Decision Tree adalah untuk membuat model pelatihan yang dapat digunakan untuk memprediksi kelas atau nilai variabel target dengan mempelajari aturan keputusan sederhana yang disimpulkan dari data sebelumnya (data pelatihan). Terdapat tiga elemen dalam satu decision tree, yaitu root node (akar) yang merupakan tujuan akhir atau keputusan besar yang ingin diambil, branches (ranting) yang merupakan berbagai pilihan tindakan, dan leaf node (daun) yang merupakan kemungkinan hasil atas setiap tindakan.

Pada algoritma yang digunakan kita perlu menginputkan library yang dibutuhkan terlebih dahulu. Selanjutnya kita perlu memasukan dataset kedalam program. Dataset yang digunakan yaitu 'data_decision_trees.txt'. Untuk memasukan dataset ke dalam program kita dapat menggunakan syntax "`np.loadtxt`". Karena Decision tree merupakan algoritma supervised learning maka kita perlu untuk melakukan pelabelan pada dataset yang akan digunakan sebagai data input pemodelan. Pada program ini kita melakukan pelabelan dataset menjadi 2, yaitu *class_0* dan *class_1*. Setelah dilakukan pelabelan kemudian dataset di plot, class_0 divisualisasikan dengan merker 'x', dan class_1 divisualisasikan dengan marker 'o'.

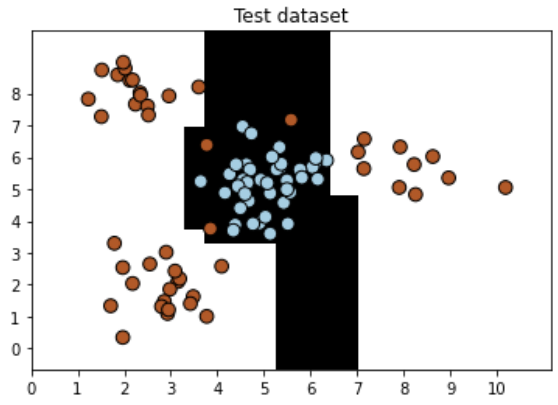


Gambar 3 Visualisasi Data Input

Setelah diberi label kemudian dataset akan di bagi menjadi data testing dan data training menggunakan syntax "`X_train, X_test, y_train, y_test`". Pemodelan pada program ini menggunakan syntax "`DecisionTreeClassifier`", yang kemudian di inputkan ke dalam variable *classifier*. Tahap selanjutnya yaitu melakukan pemodelan pada data testing dan data training. Pemodelan pada data train dilakukan menggunakan syntax "`classifier.fit(X_train, y_train)`". Sedangkan pemodelan pada data test dilakukan menggunakan syntax "`y_test_pred = classifier.predict(X_test)`". Setelah dimodelkan kita dapat melihat hasilnya menggunakan syntax "`visualize_classifier`" yang berasal dari library utilities.



Gambar 4 Pemodelan pada Training dataset



Gambar 5 Pemodelan pada Test Dataset

Selanjutnya dilakukan proses evaluasi untuk melihat keakuratan dari model yang telah dibuat. Evaluasi pada data training dapat dilakukan menggunakan syntax "`classification_report(y_train, classifier.predict(X_train))`". Adapun evaluasi pada data testing dilakukan menggunakan syntax "`classification_report(y_test, y_test_pred)`".

```
########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.99      1.00      1.00       137
     Class-1       1.00      0.99      1.00       133

    accuracy                           1.00       270
   macro avg       1.00      1.00      1.00       270
weighted avg       1.00      1.00      1.00       270

########################################
```

Gambar 6 Hasil Evaluasi Data Train

```
########################################

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.93      1.00      0.97        43
     Class-1       1.00      0.94      0.97        47

    accuracy                           0.97        90
   macro avg       0.97      0.97      0.97        90
weighted avg       0.97      0.97      0.97        90

########################################
```

Gambar 7 Hasil Evaluasi Data Test

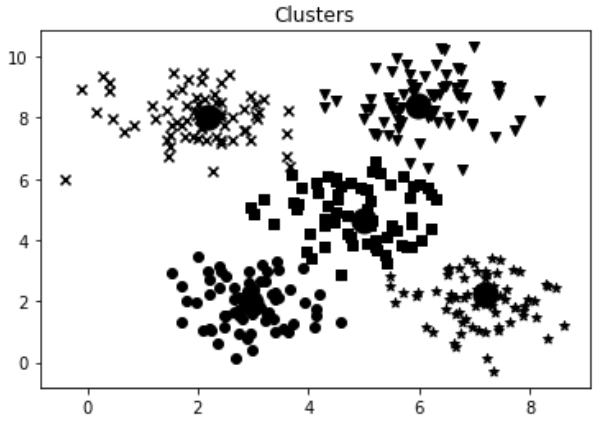
Berdasarkan hasil evaluasi, performa klasifikasi pada data train lebih baik daripada evaluasi performa pada data test. Hal ini ditunjukan dari nilai presisi dari kedua dataset, semakin mendekati 1 maka performa klasifikasi akan semakin baik.

3. Analisa mean_shift.py

Meanshift adalah algoritma unsupervised learning yang digunakan untuk melakukan clustering. Mean shift didasarkan pada konsep Kernel Density Estimation (KDE), yang merupakan cara untuk memperkirakan fungsi kepadatan probabilitas variabel acak. KDE bekerja dengan cara menempatkan kernel pada setiap titik dalam kumpulan data. Kernel adalah fungsi pembobotan pada sebuah data yang umumnya digunakan dalam konvolusi. Tergantung pada parameter bandwidth kernel yang digunakan, fungsi kepadatan yang dihasilkan akan bervariasi.

Untuk mencoba algoritma mean shift, digunakan library *numpay, matlotlib, skylearn,* dan *itertools*. Setelah di library di import ke dalam workspace selanjutnya dataset diinputkan juga. Dataset yang digunakan pada pemrograman ini yaitu "data_clustering.txt". Seperti yang dijelaskan diatas, bahwa algoritma mean shift memerlukan estimasi bandwidth untuk menghasilkan kepadatan clusterisasi yang bervariasi. Estimasi bandwidth pada program ditunjukan menggunakan syntax "`bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))`". Selanjutnya dilakukan pemodelan menggunakan bandwidth yang telah di estimasikan, syntax yang digunakan untuk melakukan pemodelan yaitu "`meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)`".

Karena algoritma Mean Shift merupakan supervised learning maka kita perlu untuk melakukan pelabelan pada dataset yang ada. Namun kelebihan algoritma ini dari algoritma sebelumnya yaitu, pada algoritma ini akan menentukan clustering terbaik secara otomatis. Jadi kita tidak perlu menerka jumlah clustering yang sesuai dengan dataset yang kita miliki. Untuk menunjukan hasil dari pemodelan dan clustering pada dataset kita perlu melakukan plotting. Ploting pada program ini menggunakan library dari matplotlib. Berikut merupakan hasil dari clustering yang telah dilakukan oleh algoritma Meanshift :



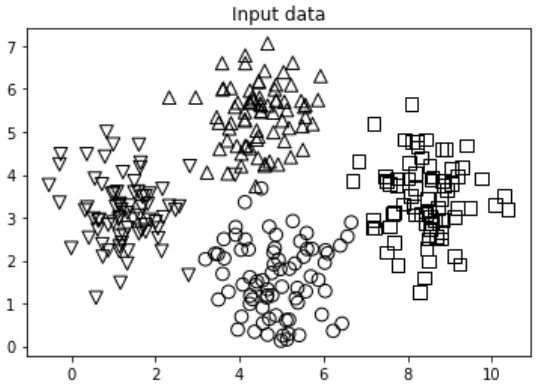
Gambar 8 Hasil Clusters Algoritma MeanShift

4. Analisa nearest_neighbors_classifier.py

KNN yang merupakan singkatan dari "K-Nearest Neighbor". K-Nearest Neighbor dilambangkan dengan "K" yang merupakan jumlah neighbor terdekat dari variabel baru yang ingin di prediksi. Algoritma KNN bekerja dengan cara mengambil sejumlah K data terdekat (neighbour) sebagai acuan untuk ,menentukan kelas dari data baru. Sederhananya algoritma knn mengklasifikasikan data berdasarkan similarity atau kedekatan terhadap data lainnya.

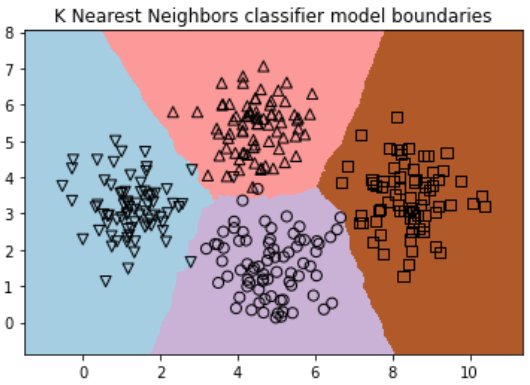
Karena KNN mengklasifikasikan data baru berdasarkan kedekatannya maka algoritma knn perlu menghitung jarak dari semua titik terdekat dari data terbaru, oleh sebab itu algoritma knn memerlukan komputasi yang besar. Untuk menghitung jarak antara dua titik data pada algoritma KNN digunakan rumus Euclidean Distance. Setelah mengetahui jarak antara dua data, kita menggunakan nilai K yang telah kita tentukan untuk menentukan neighbour dari titik data baru. Nilai K sebaiknya bernilai ganjil sehingga algoritma dapat menentukan similarity terbaik, namun kita tidak dapat menentukan nilai K dengan tepat sehingga diperlukan pengujian agar mendapatkan akurasi terbaik.

Pada program yang digunakan untuk menganalisa algoritma K-Nearest Neighbor kita perlu untuk import library yang dibutuhkan terlebih dahulu. Selanjutnya dataset dimasukkan kedalam program menggunakan syntax "`np.loadtxt`". Dataset yang telah di import kemudian ditampilkan kedalam plot menggunakan library *matplotlib* yaitu syntax *plt*. Berikut merupakan hasil visualisasi dari dataset atau data input untuk pemodelan:



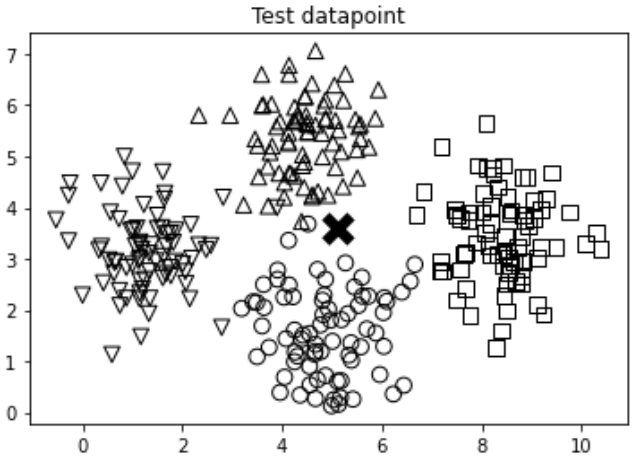
Gambar 9 Visualisasi Dataset

Selanjutnya yaitu membuat pemodelan untuk algoritma KNN. Seperti yang telah dijelaskan sebelumnya, pemodelan jenis ini kita perlu menentukan nilai "k" untuk menentukan neighbour dari titik data baru. Penentuan nilai k pada program ditunjukan menggunakan syntax "`num_neighbors = 12`". Kemudian pemodelan KNN dibuat menggunakan nilai k =12 dengan syntax "`neighbors.KNeighborsClassifier`". Agar jangkauan prediksi algoritma KNN tidak terlalu besar maka dibuat batas kemudian dilakukan ploting hasil pemodelan serta batas data algoritma KNN.



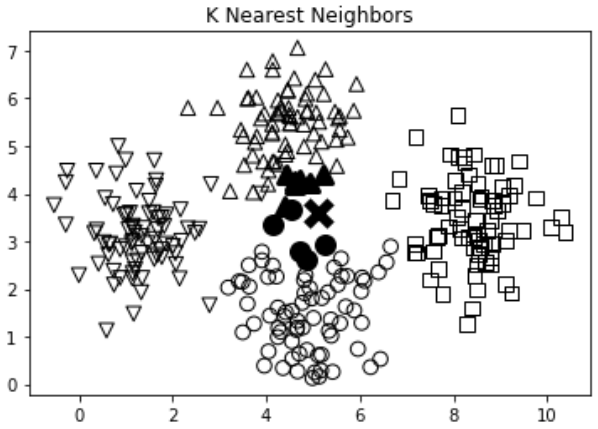
Gambar 10 Pemodelan Algoritma KNN

Kemudian kita akan melakukan evaluasi dengan cara memprediksi kedekatan dari data. Caranya yaitu dengan menentukan nilai data yang akan diprediksi, pada program digunakan syntax "`test_datapoint = [5.1, 3.6]`". Selanjutnya kita akan melakukan ploting dari data yang ingin kita prediksi dengan pemodelan KNN. Data yang akan di tes di tandai menggunakan marker "x".



Gambar 11 Plot datapoint pada pemodelan KNN

Selanjutnya, berdasarkan nilai k atau tetangga yang telah di tetapkan. Kita akan memprediksi cluster mana yang paling dekat dengan datapoint menggunakan syntax "`classifier.kneighbors([test_datapoint])`". Setelah di prediksi data mana saja yang terdekat dari datapoint, kemudian kita ploting kembali data point beserta tetangga terdekatnya.



Gambar 12 Hasil klasifikasi menggunakan algoritma KNN

Dari hasil ploting diatas kita dapat melihat bahwa ada beberapa cluster yang paling dekat dengan datapoint. Jumlah tetangga yang muncul juga telah sesuai dengan nilai k yang telah kita inputkan sebelumnya yaitu 12.

5. Analisa states.py

Program yang terakhir yaitu states.py. Program ini merupakan implementasi untuk domain NLP. Karena digunakan untuk melakukan pengolahan Bahasa. Namun sebelum running program, kita perlu untuk menginstall library logPy dari github. Setelah instalasi logPy berhasil kemudian kita akan mengimportkannya kedalam program menggunakan syntax *from logpy import run, fact, eq, Relation, var*. Pada program ini digunakan 2 buah dataset, coastal_states.txt dan adjacent_states.txt. Setelah dataset di load kedalam program, kemudian kita akan membaca dataset menggunakan syntax "*open*". Proses terakhir yaitu kita akan menginisialisasikan variabel menggunakan syntax "`x = var() & y = var()`".

```
# Is Nevada adjacent to Louisiana?
output = run(0, x, adjacent('Nevada', 'Louisiana'))
print('\nIs Nevada adjacent to Louisiana?:')
print('Yes' if len(output) else 'No')
```

```
Is Nevada adjacent to Louisiana?:
No
```

Dari program diatas, kita ingin melakukan prediksi. Apakah Nevada dan louisiana berdekatan?. Pertama-tama kita inisialisasikan dulu Nevada dan Louisiana ke dalam variable output dengan syntax "*adjacent*". Kemudian lakukan percabangan *if else* untuk menjawab pertanyaan yang diberikan. Karena negara Nevada termasuk ke dalam dataset

adjacent_state.txt dan negara Louisiana masuk kedalam dataset *coastal_states.txt* maka program menjawab 'NO'.

```
# States adjacent to Oregon
output = run(0, x, adjacent('Oregon', x))
print('\nList of states adjacent to Oregon:')
for item in output:
    print(item)
```

```
List of states adjacent to Oregon:
California
Nevada
Idaho
Washington
```

Program diatas meminta kita untuk menampilkan negara yang dekat dengan Oregon. Hal ini mudah dijawab oleh program karena pada dataset *adjacent_state.txt,* Setiap negara sudah ditampilkan berdasarkan kedekatannya. Oleh karena itu kita hanya perlu menginisialisasi Oregon pada variable output.

```
# States adjacent to Mississippi that are coastal
output = run(0, x, adjacent('Mississippi', x), coastal(x))
print('\nList of coastal states adjacent to Mississippi:')
for item in output:
    print(item)
```

```
List of coastal states adjacent to Mississippi:
Louisiana
Alabama
```

Program ini juga tidak jauh beda dengan sebelumnya. Karena dataset telah diurutkan berdasarkan kedekatannya. Maka kita hanya perlu memanggil dataset yang paling dengan dengan Mississippi.

```
# List of 'n' states that border a coastal state
n = 7
output = run(n, x, coastal(y), adjacent(x, y))
print('\nList of ' + str(n) + ' states that border a coastal state:')
for item in output:
    print(item)
```

```
List of 7 states that border a coastal state:
Georgia
Arizona
Connecticut
New York
Idaho
Rhode Island
Tennessee
```

Program diatas hanya sedikit berbeda jika dibandingkan dengan program sebelumnya. Perbedaan nya terletak pada jumlah data yang ingin ditampilkan sudah di inisialisasikan sebelumya. Inisialisasi jumlah data dilakukan menggunakan syntax “n=7”, artinya kita akan menampilkan 7 buah data yang paling dekat.

```
# List of states that adjacent to the two given states
output = run(0, x, adjacent('Arkansas', x), adjacent('Kentucky', x))
print('\nList of states that are adjacent to Arkansas and Kentucky:')
for item in output:
    print(item)
```

```
List of states that are adjacent to Arkansas and Kentucky:
Missouri
Tennessee
```

Program diatas kita diminta untuk menampilkan data paling depat dengan Arkansas dan Kentucky. Hal yang perlu dilakukan hanya menggabungkan Arkansas dan kentucky kedalam sebuah fungsi yang sama yaitu “`output = run(0, x, adjacent('Arkansas', x), adjacent('Kentucky', x))`”.