Introduction to Data Science HS 2021

Aufgabenblatt 2: *k*-Nächste-Nachbarn und *k*-Mitten

Die Bearbeitung der Aufgaben ist freiwillig; es erfolgt keine Bewertung.

Aufgabe 1

a) Gegeben ist folgende Tabelle mit $n = 12$ Werten der tatsächlichen Klassifikation y und der durch k-Nächste-Nachbarn vorhergesagten Klassifikation $\hat{y}$ eines Datensatzes (Klassen 0 und 1). Bestimmen Sie die Treffsicherheit des Algorithmus.

i	1	2	3	4	5	6	7	8	9	10	11	12
y_i	1	1	1	1	1	1	0	1	0	1	1	0
$\hat{y}_i$	1	1	1	1	0	1	1	0	0	1	0	0

– Lösung –

- Treffsicherheit: $\frac{1}{n}\sum_i 1(\hat{y}_i = y_i)$
- Für $n = 12$ folgt aus obiger Tabelle bei acht Übereinstimmungen ($i = 1, 2, 3, 4, 6, 9, 10, 12$) und vier Nicht-Übereinstimmungen ($i = 5, 7, 8, 11$), dass die Treffsicherheit gleich $8/12 = 0.667$ oder 66.7 % beträgt.

b) Wie gross ist demnach die Fehlinterpretationsrate (Hamming-Verlust)?

– Lösung –

- Verlust ist 1 – Treffsicherheit $\Rightarrow$ Verlust ist $1 - 0.667 = 0.333$ oder 33.3 %.

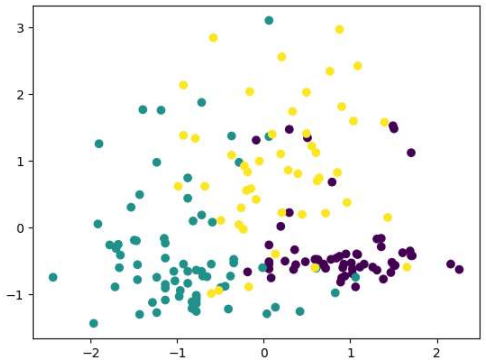
Aufgabe 2

Die Datei „`wine.txt`“ in Moodle enthält drei Spalten des zugehörigen Datensatzes aus dem Paket `sklearn.datasets`. Die ersten beiden Spalten entsprechen 2D-Koordinaten (x und y), die dritte Spalte entspricht der tatsächlichen Klassifikation der Punkte.

a) Visualisieren Sie die Datei als Streudiagramm (Scatter Plot).

– Lösung –

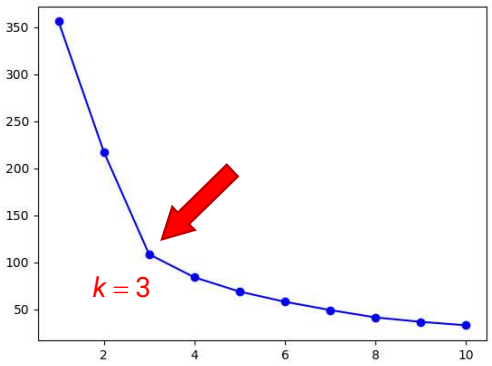
```
>>> data = np.loadtxt("wine.txt", delimiter=",")
>>> plt.scatter(data[:,0], data[:,1], c=data[:,2]
>>> plt.show()
```



b) Ermitteln Sie anhand der Ellenbogen-Methode den optimalen *k*-Wert für eine Clusterbildung gemäss *k*-Mitten-Algorithmus und zeichnen Sie die zugehörige (Trägheits)Kurve.

– Lösung –

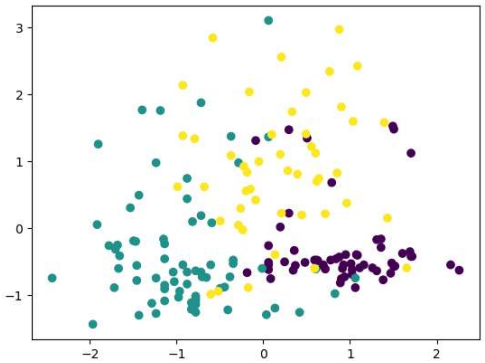
```
>>> x_data = data[:,:2]    # Spalten 0 und 1 → Koordinaten
>>> y_data = data[:,2]     # Spalte 2 → korrekte Klassifikation
>>> inert = []
>>> for k in range(1,11):
        model = KMeans(k)
        model.fit(x_data)
        inert.append(model.inertia_)
>>> x = np.linspace(1,10,10)
>>> plt.plot(x, inert, 'b-')
>>> plt.plot(x, inert, 'bo')
>>> plt.show()
```



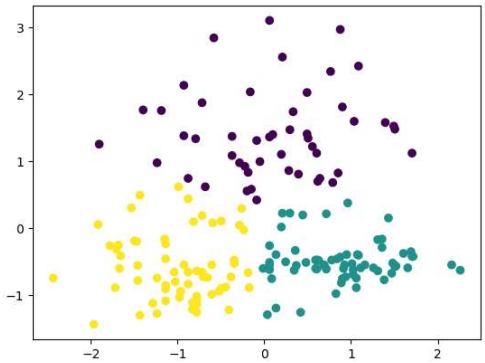
c) Bestimmen Sie für den aus b) ermittelten optimalen *k*-Wert eine Vorhersage (**`fit_predict`**) der Klassenzugehörigkeit aller Punkte der Datei „**`wine.txt`**“ und visualisieren Sie das Ergebnis als Streudiagramm. Wie gut stimmt die vorhergesagte Klassifikation – rein optisch betrachtet – mit dem tatsächlichen Ergebnis aus a) überein?

– Lösung –

```
>>> model = KMeans(3)
>>> y_pred = model.fit_predict(x_data)
>>> plt.scatter(data[:,0], data[:,1], c=y_pred)
>>> plt.show()
```



tatsächliche Klassifikation aus a)



vorhergesagte Klassifikation ($k = 3$)

Rein optisch betrachtet, ist die Klassifikation durchaus als gut zu bewerten (die unterschiedliche Farbgebung ist irrelevant). Einzelne Punkte, die zu weit vom eigentlichen Cluster entfernt liegen, wurden zwar falsch klassifiziert, aber das ist bei diesem Verfahren auch nicht anders zu erwarten.

Aufgabe 3 (für Experimentierfreudige)

In der Vorlesung wurde für die Datei „**`smp_data.txt`**“ (*x*-*y*-Punktkoordinaten) mittels linearer Regression die zugehörige Regressionsgerade bestimmt. Das Paket **`sklearn.neighbors`** bietet hier die Möglichkeit, mittels **`KNeighborsRegressor`** eine Regression basierend auf dem bekannten *k*-Nächste-Nachbarn-Algorithmus durchzuführen. Dabei wird das zugehörige Modell mit den *x*-Punktkoordinaten als Daten und den *y*-Punktkoordinaten als Klassifikation trainiert.

```
>>> data = np.loadtxt("smp_data.txt", delimiter=",")
>>> x_data = data[:,0].reshape((-1,1))
>>> y_data = data[:,1]
>>> from sklearn.neighbors import KNeighborsRegressor as knr
>>> model = knr()
>>> model.fit(x_data, y_data)
```

a) Bestimmen Sie den optimalen *k*-Wert der Regression gemäss **`KNeighborsRegressor`** für die Daten aus der Datei „**`smp_data.txt`**“. Beachten Sie, dass es sich um eine Regression und nicht um eine Klassifizierung handelt, d.h. die Qualität der Vorhersage durch **`KNeighborsRegressor.predict()`** muss mittels R^2-Wert bestimmt werden.

– Lösung –

```
>>> from sklearn.metrics import r2_score
>>> for k in range(1,16):
        model = knr(k)
        model.fit(x_data, y_data)
        y_pred = model.predict(x_data)
        print(k, ":", r2_score(y_data, y_pred))
```

Das beste Ergebnis mit 0.7954526029533308 wird für $k = 5$ erreicht.

b) Erzeugen Sie sich eine Sequenz aus 50 *x*-Punktkoordinaten im Bereich [1, 11] mittels

```
>>> x = np.linspace(1,11,50)
```

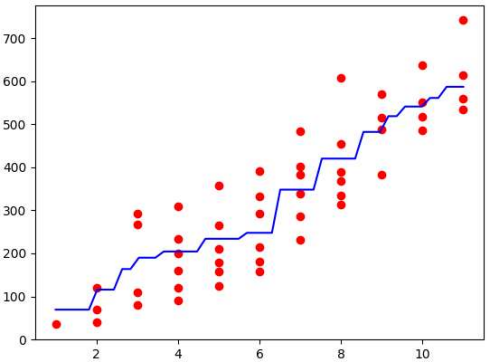
und nutzen Sie diese Sequenz für die Vorhersage durch **`KNeighborsRegressor`** unter Verwendung des in Aufgabe a) ermittelten, optimalen Wertes für *k*.

```
>>> model = knr( --Ihr-k-Wert-- )
>>> model.fit(x_data, y_data)
>>> y = model.predict(x.reshape((-1,1)))
```

Die vorhergesagten Werte entsprechen den zugehörigen *y*-Punktkoordinaten der Regressionskurve. Zeichnen Sie die Originaldaten inklusive Regressionskurve in einem Diagramm.

– Lösung –

```
>>> plt.plot(x_data, y_data, 'ro')
>>> plt.plot(x, y, 'b-')
>>> plt.show()
```


700
600
500
400
300
200
100
0
2
4
6
8
10