Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №2

Програмування ітелектуальних інформаційних систем

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірила:
Баришич Л. М

Київ 2023

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Класифікація даних.

2 ЗАВДАННЯ

Метрики і спосіб виконання описані тут:
https://www.kaggle.com/code/prashant111/naive-bayes-classifier-in-python

Лабу можна виконати в онлайн-редакторах типу Google Collab.

1. Dataset1: /kaggle/input/adult-dataset/adult.csv'

Bayesian Classification + Support Vector Machine

Зробити предікшн двома вищезгаданими алгоритмами. Порівняти наступні метрики:

Recall, f1-score, Confusion matrix, accuracy score. Порівняти з нуль-гіпотезою і перевірити на оверфітинг. Пояснити результати.

2. Dataset2: https://www.kaggle.com/code/stieranka/k-nearest-neighbors

K nearest neighbours.

Те саме що і в 1 завданні, але порівнюємо між собою метрики. Euclidean, Manhattan, Minkowski. Кластери потрібно візуалізувати. Метрики аналогічно п.1

3. Dataset3: https://www.kaggle.com/code/nuhashafnan/cluster-analysis-kmeans-kmediod-agnes-birch-dbscan

Agnes,Birch,DBSCAN

Інші методи можна ігнорувати. Зняти метрики (Silhouette Coefficient, ARI, NMI. Можна з п.1-2), пояснити.

4. Dataset4: https://www.kaggle.com/code/datark1/customers-clustering-k-means-dbscan-and-ap

Affinity propagation.

Порівняти з k-means. Метрики - Silhouette Coefficient, ARI, NMI

У звіті до кожної задачі:

1 Візуалізувати кластери

2 Вивести метрики. Для кластерів - Silhouette Coefficient, ARI, NMI

3 Порівняння з нулем і перевірка на оверфіт.

4 Висновок.

SVM і AP можна виконати на будь-якому датасеті.

3 ВИКОНАННЯ

3.1 Завдання друге

Для початку імпортуємо модулі. Завантажимо датафрейм та виведемо його вміст.

In [53]:
```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import matplotlib.ticker as ticker
from sklearn import preprocessing
df = pd.read_csv('data/teleCust1000t.csv')
df
```

Out[53]:

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	cust
0	2	13	44	1	9	64.0	4	5	0.0	0	2	
1	3	11	33	1	7	136.0	5	5	0.0	0	6	
2	3	68	52	1	24	116.0	1	29	0.0	1	2	
3	2	33	33	0	12	33.0	2	0	0.0	1	1	
4	2	23	30	1	9	30.0	1	2	0.0	0	4	
...	...	...	...	...	...	...	...	...	...	...	...	
995	3	10	39	0	0	27.0	3	0	0.0	1	3	
996	1	7	34	0	2	22.0	5	5	0.0	1	1	
997	3	67	59	0	40	944.0	5	33	0.0	1	1	
998	3	70	49	0	18	87.0	2	22	0.0	1	1	
999	3	50	36	1	7	39.0	3	3	0.0	1	3	

1000 rows × 12 columns

Рисунок 3.1.1 - Сутності

Перетворимо датафрейм у масив NumPy.

In [54]:
```
X = df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside']].values
y = df['custcat'].values
```

Рисунок 3.1.2 - Перетворення датафрейму у масив NumPy

Нормалізуємо дані

In [55]:
```
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X
```

Out[55]:
```
array([[-0.02696767, -1.055125  ,  0.18450456, ..., -0.22207644,
        -1.03459817, -0.23065004],
       [ 1.19883553, -1.14880563, -0.69181243, ..., -0.22207644,
        -1.03459817,  2.55666158],
       [ 1.19883553,  1.52109247,  0.82182601, ..., -0.22207644,
         0.96655883, -0.23065004],
       ...,
       [ 1.19883553,  1.47425216,  1.37948227, ..., -0.22207644,
         0.96655883, -0.92747794],
       [ 1.19883553,  1.61477311,  0.58283046, ..., -0.22207644,
         0.96655883, -0.92747794],
       [ 1.19883553,  0.67796676, -0.45281689, ..., -0.22207644,
         0.96655883,  0.46617787]])
```

Рисунок 3.1.3 - Нормалізація даних

Натренуємо модель.

```
In [57]: from sklearn.neighbors import KNeighborsClassifier
         k = 4
         neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
         neigh

Out[57]: KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=4)
```

Рисунок 3.1.4 - Тренування моделі

Класифікуємо дані.

```
In [58]: yhat = neigh.predict(X_test)
         yhat[0:5]

Out[58]: array([1, 1, 3, 2, 4])
```

Рисунок 3.1.5 - Класифікуємо дані

Перевіримо точність.

```
In [59]: from sklearn import metrics
         print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
         print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))

Train set Accuracy:  0.5475
Test set Accuracy:  0.32
```

Рисунок 3.1.6 - Точність

3.2 Minkowski

Бачимо, що точність замала. Спробуємо підіюрати інші значення k. Визначимо, які варіанти параметрів найкраще вирішують дану задачу для різних дистанцій.

Визначимо найкращі параметри для Minkowski.

```
In [61]: from sklearn.model_selection import GridSearchCV
         classificator = KNeighborsClassifier(metric='minkowski')
         params = {'n_neighbors': range(1, 60)}
         grid_search = GridSearchCV(classificator, params, cv=10, verbose=1)
         grid_search.fit(X_train, y_train)
         knn = grid_search.best_estimator_
         knn

Fitting 10 folds for each of 59 candidates, totalling 590 fits

Out[61]: KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=19)
```

Рисунок 3.2.1 - Визначення найкращого параметра

Натренуємо модель з найкращим параметром.

```
In [62]: knn.fit(X_train, y_train)

Out[62]: KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=19)
```

Рисунок 3.2.2 - Тренування моделі K-Nearest Neighbors

Визначимо точність моделі на тренувальних та тестових даних. Бачимо, що модель має схильність до оверфітингу.

```
In [63]: y_pred = knn.predict(X_test)
         train_score = round(knn.score(X_train, y_train), 5)
         test_score = round(knn.score(X_test, y_test), 5)
         print(f'Train accuracy: {train_score}')
         print(f'Test accuracy: {test_score}')

Train accuracy: 0.45125
Test accuracy: 0.345
```

Рисунок 3.2.3 - Точність моделі K-Nearest Neighbors

Перевірка з нульовою точністю.

```
In [64]: def calc_null_accuracy(y_train):
             y_pd = pd.DataFrame(y_train)
             dd = y_pd.value_counts().values
             return dd[0] / (sum(dd))
         null_acc = calc_null_accuracy(y_train)
         null_acc

Out[64]: 0.28375
```

Рисунок 3.2.4 - Перевірка з нульовою точністю

Матриця невідповідностей

```
In [65]: from sklearn.metrics import confusion_matrix
         import seaborn as sns
         def conf_mat(model, x_test, y_test):
             y_predicted = model.predict(x_test)
             cm = confusion_matrix(y_test, y_predicted)
             plt.figure(figsize = (8,5))
             sns.heatmap(cm, annot=True, fmt=".1f")
             plt.xlabel('Predicted')
         conf_mat(knn, X_test, y_test)
```

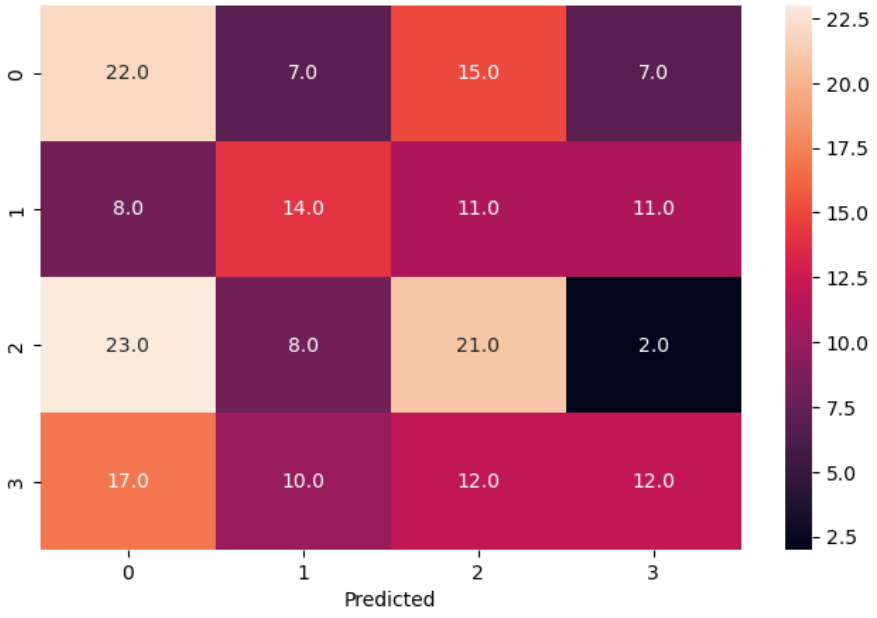

Рисунок 3.2.5 - Матриця невідповідностей

Визначимо метрики класифікації

```
In [66]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.31      0.43      0.36        51
           2       0.36      0.32      0.34        44
           3       0.36      0.39      0.37        54
           4       0.38      0.24      0.29        51

    accuracy                           0.34       200
   macro avg       0.35      0.34      0.34       200
weighted avg       0.35      0.34      0.34       200
```

Рисунок 3.2.6 - Метрики класифікації

І в кінці побудуємо матрицю кореляцій.

```
In [67]: def corr_map(df, figsize):
             fig, axis = plt.subplots(figsize=figsize)
             axis.set_title('Кореляція між факторами')
             sns.heatmap(df.corr(), ax=axis, annot=True)
         corr_map(df, (10, 6))
```

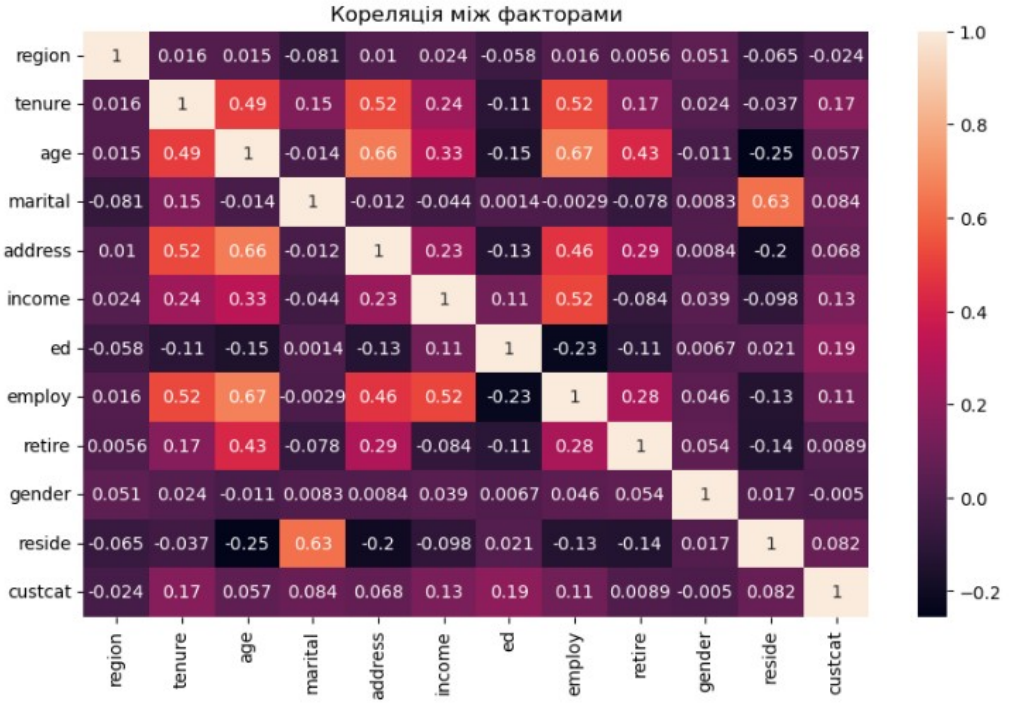

Рисунок 3.2.7 - Матриця кореляцій

Візуалізуємо спрогнозовані класи відносно tenure та ed.

```
In [73]: X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
         plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40, edgecolor="k")
         plt.show()
```

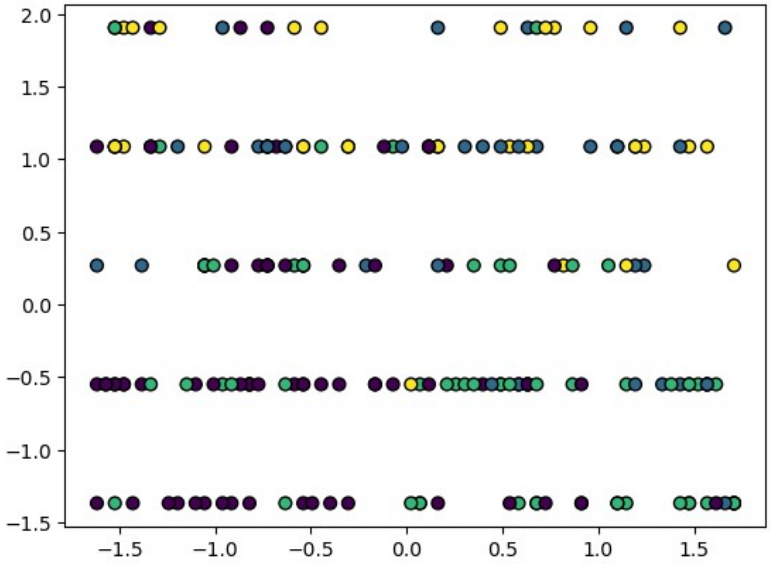

Рисунок 3.2.8 - Класи

3.3Euclidian, Manhattan

Визначимо найкращі параметри для Euclidian.

```
In [ ]: from sklearn.model_selection import GridSearchCV
        models = []
        for d in ['euclidean', 'manhattan']:
            classificator = KNeighborsClassifier(metric='euclidean')
            params = {'n_neighbors': range(1, 60)}
            grid_search = GridSearchCV(classificator, params, cv=10, verbose=1)
            grid_search.fit(X_train, y_train)
            mod = grid_search.best_estimator_
            mod.fit(X_train, y_train)
            y_pred = mod.predict(X_test)
            train_score = round(mod.score(X_train, y_train), 5)
            test_score = round(mod.score(X_test, y_test), 5)
            print(f'Train accuracy {d}: {train_score}')
            print(f'Test accuracy {d}: {test_score}')
            models.append(mod)
```

```
Fitting 10 folds for each of 59 candidates, totalling 590 fits
Train accuracy euclidean: 0.45125
Test accuracy euclidean: 0.345
Fitting 10 folds for each of 59 candidates, totalling 590 fits
Train accuracy manhattan: 0.45125
Test accuracy manhattan: 0.345
```

Рисунок 3.3.1 - Матриця кореляцій

Візуалізуємо спрогнозовані класи відносно tenure та ed для Euclidian.

```
In [74]: y_pred = models[0].predict(X_test)
         X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
         plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40, edgecolor="k")
         plt.show()
```

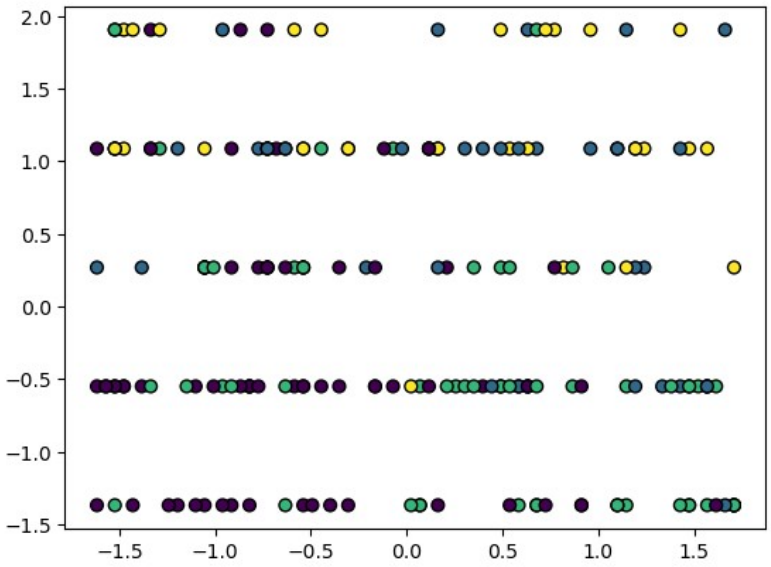

Рисунок 3.3.2 - класи Euclidian

Візуалізуємо спрогнозовані класи відносно tenure та ed для Manhattan.

```
In [75]: y_pred = models[1].predict(X_test)
         X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
         plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40, edgecolor="k")
         plt.show()
```

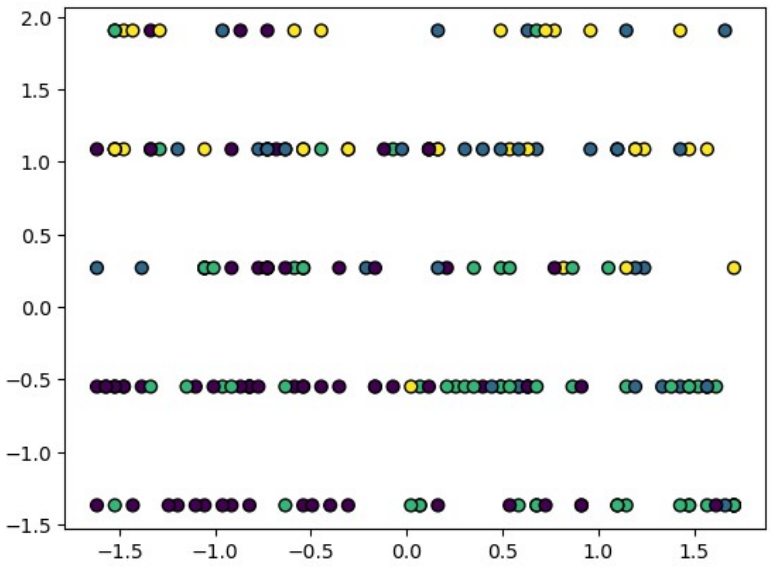

Рисунок 3.3.3 - класи Manhattan

3.4Пояснення результатів

Отже, кінцева точність для всіх трьох відстаней складає 0.345, перевірка з нульовою точністю показує результат 0.28375, а тому модель показує хоч і погану роботу у прогнозуванні класів, але точність вища за нульову.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

_____

(Обсяг програми (документа), арк.)

Студента групи ІП-113 курсу

Панченка С. В

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import matplotlib.ticker as ticker
from sklearn import preprocessing
df = pd.read_csv('data/teleCust1000t.csv')
df
X = df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
'employ', 'retire', 'gender', 'reside']].values
y = df['custcat'].values
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,
random_state=4)
from sklearn.neighbors import KNeighborsClassifier
k = 4
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
yhat = neigh.predict(X_test)
yhat[0:5]
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train,
neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
from sklearn.model_selection import GridSearchCV
classificator = KNeighborsClassifier(metric='minkowski')
params = {'n_neighbors': range(1, 60)}
grid_search = GridSearchCV(classificator, params, cv=10, verbose=1)
grid_search.fit(X_train, y_train)
knn = grid_search.best_estimator_
knn
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
train_score = round(knn.score(X_train, y_train), 5)
test_score = round(knn.score(X_test, y_test), 5)
print(f'Train accuracy: {train_score}')
print(f'Test accuracy: {test_score}')
def calc_null_accuracy(y_train):
y_pd = pd.DataFrame(y_train)
```

```
      dd = y_pd.value_counts().values
      return dd[0] / (sum(dd))
      null_acc = calc_null_accuracy(y_train)
      null_acc
      from sklearn.metrics import confusion_matrix
      import seaborn as sns
      def conf_mat(model, x_test, y_test):
      y_predicted = model.predict(x_test)
      cm = confusion_matrix(y_test, y_predicted)
      plt.figure(figsize = (8,5))
      sns.heatmap(cm, annot=True, fmt=".1f")
      plt.xlabel('Predicted')
      conf_mat(knn, X_test, y_test)
      from sklearn.metrics import classification_report
      print(classification_report(y_test, y_pred))
      def corr_map(df, figsize):
      fig, axis = plt.subplots(figsize=figsize)
      axis.set_title('Кореляція між факторами')
      sns.heatmap(df.corr(), ax=axis, annot=True)
      corr_map(df, (10, 6))
      X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
      plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40,
edgecolor="k")
      plt.show()
      from sklearn.model_selection import GridSearchCV
      models = []
      for d in ['euclidean', 'manhattan']:
      classificator = KNeighborsClassifier(metric='euclidean')
      params = {'n_neighbors': range(1, 60)}
      grid_search = GridSearchCV(classificator, params, cv=10, verbose=1)
      grid_search.fit(X_train, y_train)
      mod = grid_search.best_estimator_
      mod.fit(X_train, y_train)
      y_pred = mod.predict(X_test)
      train_score = round(mod.score(X_train, y_train), 5)
      test_score = round(mod.score(X_test, y_test), 5)
      print(f'Train accuracy {d}: {train_score}')
      print(f'Test accuracy {d}: {test_score}')
      models.append(mod)
      y_pred = models[0].predict(X_test)
      X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
      plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40,
edgecolor="k")
      plt.show()
```

```
      y_pred = models[1].predict(X_test)
      X_test_pd = pd.DataFrame(X_test, columns=df.columns[:-1])
      plt.scatter(X_test_pd.tenure, X_test_pd.ed, marker="o", c=y_pred, s=40,
edgecolor="k")
      plt.show()
```