Scientific report –Classification

ליעד בן יחיאל – 207637414

קישור למחברת פייתון -
https://colab.research.google.com/drive/17IcYMLxRMDy60mzLpH7Sx3FMR-Q6TPQy?usp=sharing

שלב 0 – ייבוא ספריות

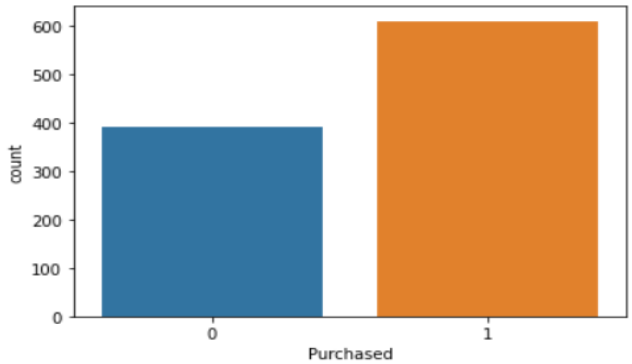
```
from sklearn.svm import SVC
from sklearn.metrics import f1_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd
import seaborn as sns
from imblearn.over_sampling import SMOTE
```

שלב 1 – חיתוך משתנים

```
data = pd.read_csv('dataset.csv') #read the data to pd
X = data.iloc[:,:-1].values #slice X features
y = data.iloc[:,-1].values #slice wanted y to predict
```

שלב 2 – התפלגות הנתונים

לאחר בדיקה נראה שהנתונים אינם מתפלגים די זהה (כפי שניתן לראות בגרף), ונבדקה האפשרות לאיזון הדאטה בשיטות שונות. לבסוף נבחרה שיטת SMOTE.




```
sns.countplot(data['Purchased']) #data doesn't has balance.
```

שיטה זו מייצרת נתונים סינתטיים עבור הקבוצה הקטנה (Purchased = 0) פועלת על ידי בחירה אקראית של נקודה מהקבוצה הקטנה ומחשבת את השכנים הקרובים ביותר K עבור נקודה זו. הנקודות הסינתטיות מתווספות בין הנקודה הנבחרת לשכנותיה.

Balance The data by using Synthetic Minority Oversampling Technique (SMOTE)

```
[94] smote = SMOTE()
     X,y = smote.fit_resample(X, y) #fit predictor and target variable
```

שלב 3 – סקיילינג

ניתן לראות שישנם רק 2 משתנים בלתי תלויים: גובה, וגודל משכורת אשר מתפלגים בצורה שונה ולכן היה חשוב לבצע ביניהם סקיילינג.

```
mm_x = MinMaxScaler() #for scaling the data
```

היישום עצמו יתבצע כל פעם על הTrain set ולא על הדאטה כולו כפי שנלמד.

שלב 4 – יצירת פונקציה להדפסת הפרמטרים וF1

```
def measuring_metrics(f1,checked_parmeter):
    result_table={}
    result_table["checked parmeter"] = np.array(checked_parmeter)
    result_table["f1 mean score"] = np.array(np.mean(f1, axis=0)) #calculate f1 mean
    result_table["f1 std score"] = np.array(np.std(f1, axis=0)) #calculate f1 std
    result_table = pd.DataFrame(result_table)
    search_best_param = np.mean(f1, axis=0).argmax() #lock the best param
    print(result_table)
    print("The best parameter accroding to f1 measure metric is: " + str(checked_parmeter[search_best_param]))
```

שלב 5 – הרצת המודלים

- בכל מודל נבנה ווקטור אפסים התחלתי במדדים על פי מספר הפרמטרים הנבדקים.
- רצה לולאה של 1000 שמפצלת ל Train set ו Test set בגודל של 10% מהData. אשר מפצלת כל פעם לפי random state שונה 1-1000 כפי שהתבקש.
- בתוך לולאה זו רצה לולאה אשר בודקת על פי המתבקש, עושה סקיילינג, מאמנת את המודל ומחזירה כמתבקש (יפורט על פי כל אחד מהמודלים).

שלב 5.1 - K-nn (based on Euclidian distance)

```
def K_nn(X,y):
    f1 = np.zeros((1000,20))
    for state_num in range(0,1000,1):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=state_num)
        for k_checked in range(1,21,1):
            X_train = mm_x.fit_transform(X_train)
            X_test = mm_x.fit_transform(X_test)
            model = KNeighborsClassifier(n_neighbors=k_checked ,p=2)
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            f1[state_num,k_checked-1]=f1_score(y_test, y_pred, average=None)[1]
    measuring_metrics(f1,np.arange(1,21,1))
    return
print('K-NN metrics results:')
K_nn(X,y)
```

כפי שסומן בלולאה הפנימית נבדקים היפרפרמטריי K בגודל 1-20.

על כל היפרפרמטר המודל מאומן ומוחזר F1 ממוצע וסטיית התקן שלו. הK המיטבי נבחר על פי F1 הממוצע הגדול ביותר ולבסוף מודפסת הטבלה הכוללת את כלל הבדיקות והK המיטבי.

K-NN metrics results:

	checked parmeter	f1 mean score	f1 std score
0	1	0.884965	0.030504
1	2	0.881413	0.030898
2	3	0.899943	0.029161
3	4	0.903108	0.029190
4	5	0.910485	0.027917
5	6	0.906931	0.028776
6	7	0.909568	0.028121
7	8	0.903679	0.029398
8	9	0.905601	0.028266
9	10	0.898457	0.029249
10	11	0.900501	0.028643
11	12	0.894545	0.029687
12	13	0.897841	0.028819
13	14	0.891898	0.029959
14	15	0.895418	0.029523
15	16	0.889487	0.030400
16	17	0.891313	0.030209
17	18	0.885709	0.030774
18	19	0.886389	0.030819
19	20	0.882731	0.031297

The best parameter accroding to f1 measure metric is: 5

שלב 5.2 - Logistic regression

```
def logistic_regression(X,y):
    f1= np.zeros((1000,1))
    for state_num in range(0,1000,1):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = state_num)
        X_train = mm_x.fit_transform(X_train)
        X_test = mm_x.fit_transform(X_test)
        LR = LogisticRegression()
        model = LR.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        f1[state_num]=f1_score(y_test, y_pred, average=None)[1]
    print ("f1 mean score is: " + '\t' + str(np.mean(f1)))
    print ("f1 std score is: " +  '\t' + str(np.std(f1)))
    return
print('Logistic regression metrics results:')
logistic_regression(X,y)
```

במודל זה אין הייפרמטרים נבדקים ולכן המודל רץ 1000 פעמים בפיצולים רנדומאליים שונים ולבסוף מוחזר F1 ממוצע וסטיית התקן.

```
Logistic regression metrics results:
f1 mean score is:       0.7554257514667392
f1 std score is:        0.04250016990959811
```

שלב 5.3 - Linear SVC

```
def linear_SVC(X ,y):
    f1= np.zeros((1000))
    for state_num in range(0,1000,1):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = state_num)
        X_train = mm_x.fit_transform(X_train)
        X_test = mm_x.fit_transform(X_test)
        model = SVC(kernel ="linear")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        f1[state_num]=f1_score(y_test, y_pred, average=None)[1]
    print( "f1 mean score is:" + '\t' + str(np.mean(f1)))
    print( "f1 std score is:" + '\t' + str(np.std(f1)))
    return
print('Linear SVC metrics results:')
linear_SVC(X,y)
```

כנ"ל לא נבדקים במודל זה היפרפרמטרים ולכן גם כאן המודל רץ 1000 פעמים בפיצולים רנדומאליים שונים ולבסוף מוחזר F1 ממוצע וסטיית התקן.

```
Linear SVC metrics results:
f1 mean score is:       0.7575454548087536
f1 std score is:        0.042433175327916635
```

שלב 5.4 - Polynomial SVC

במודל זה אכן נבדקים היפרפרמטרים: דרגות 2-5 כפי שמסומן באדום.

```
def Polynomial_SVC(X,y):
    f1= np.zeros((1000,4))
    for state_num in range(0,1000,1):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = state_num)
        X_train = mm_x.fit_transform(X_train)
        X_test = mm_x.fit_transform(X_test)
        for deg in range(2,6,1):
            model = SVC(kernel ="poly", degree = deg)
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            f1[state_num,deg-2]= f1_score(y_test, y_pred, average=None)[1]
    measuring_metrics(f1,np.arange(2,6,1))
    return
print('Polynomial SVC metrics results:')
Polynomial_SVC(X,y)
```

גם כאן המודל רץ 1000 פעמים על פיצולים רנדומאליים שונים על כל היפרפרמטר המודל מאומן ומוחזר F1 ממוצע וסטיית התקן שלו. הדרגה המיטבית נבחרת על פי F1 הממוצע הגדול ביותר ולבסוף מודפסת הטבלה הכוללת את כלל הבדיקות והדרגה המיטבית.

```
Polynomial SVC metrics results:
   checked parmeter  f1 mean score  f1 std score
0                 2       0.756127      0.043728
1                 3       0.751963      0.044825
2                 4       0.755933      0.044908
3                 5       0.761205      0.043758
The best parameter accroding to f1 measure metric is: 5
```

שלב 5.5 - Gaussian SVC

במודל זה אכן נבדקים היפרפרמטרים המתבקשים:

C בגדלים - 0.2 , 0.5 , 1.2 ,1.8 ,3 כפי שמסומן באדום.

```
def Gaussian_SVC(X,y):
    f1= np.zeros((1000,5))
    for state_num in range(0,1000,1):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = state_num)
        X_train = mm_x.fit_transform(X_train)
        X_test = mm_x.fit_transform(X_test)
        for c in (0.2 , 0.5 , 1.2 ,1.8 ,3):
            model = SVC(C=c) #rbf as defualt
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            f1[state_num,(0.2 , 0.5 , 1.2 ,1.8 ,3).index(c)]= f1_score(y_test, y_pred, average=None)[1]
    measuring_metrics(f1,(0.2 , 0.5 , 1.2 ,1.8 ,3))
    return
print('Gaussian SVC metrics results:')
Gaussian_SVC(X,y)
```

גם כאן המודל רץ 1000 פעמים על פיצולים רנדומאליים שונים על כל היפרפרמטר המודל מאומן ומוחזר F1 ממוצע וסטיית התקן שלו. הC המיטבי נבחר על פי F1 הממוצע הגדול ביותר ולבסוף מודפסת הטבלה הכוללת את כלל הבדיקות והC המיטבי.

```
Gaussian SVC metrics results:
   checked parmeter  f1 mean score  f1 std score
0               0.2       0.844165      0.035090
1               0.5       0.865536      0.032464
2               1.2       0.875636      0.031463
3               1.8       0.879262      0.030893
4               3.0       0.886652      0.030726
The best parameter accroding to f1 measure metric is: 3
```