הסבר וורד מטלה 3 – COVID-19

ליעד בן יחיאל – 207637414

שלב 0 – שאיבת הדאטה

כשלב ראשוני נכנסתי לקובץ הCVS שצורף ובעקבות הבנת משמעות העמודות וההוראות המתבקשות החלטתי איזה עמודות לקחת כמשתנים בלתי תלויים ואילו משתנים כמשתנים תלויים.

```
df = pd.read_csv('COVID-19_Daily_Testing_-_By_Test.csv')
X = df.iloc[:, 5:23].values
y_pos = df.iloc[:, 2].values
y_neg = df.iloc[:, 3].values
```

שלב 1 – ניקוי הדאטה

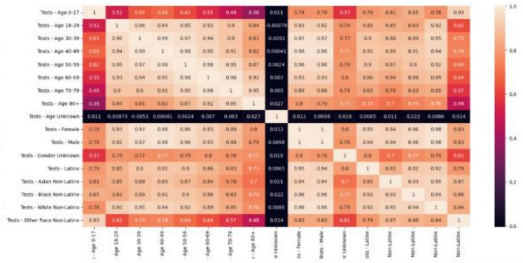
כחלק ראשוני לאחר קביעת המשתנים התלויים והבלתי תלויים למדנו שעלינו לנקות את הדאטה. בעקבות צפייה על הדאטה ובדיקה נצפה שישנם עמודות ושורות עם ערכים נורא גבוהים מכיוון שהיו באותו יום בדיקות רבות וימים עם ערכים נורא נמוכים.

ולכן נערך סקיילינג לכלל הנתונים כפי שנלמד.

```
mm_x = MinMaxScaler()
X_train = mm_x.fit_transform(X_train)
X_test  = mm_x.transform(X_test)
```

כמו התבצע ניקוי לפיצ'ארים שביניהם ניכרת קורולצייה גבוהה (0.975מ).

החקירה על הקורולציה התבצעה רק על הTRAIN SET אך הושמטו גם העמודות בהתאם בTEST SET על מנת לשמור על אותו גודל מערך נתונים.



בעקבות מפת קורולציה שבה נצפה לרוב קורלציות גבוהות שהורידו פיצ'ארים רבים מידיי נקבע נקודת החלטה לקורולצייה גבוהה מ0.975

```
def clean_high_correlation(X_train,X_test):
    df_temp_train = pd.DataFrame(X_train)
    df_temp_test = pd.DataFrame(X_test)
    corr_matrix = df_temp_train.corr().abs() # Create correlation matrix
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
    to_drop = [column for column in upper.columns if any(upper[column] > 0.975)] # Find
    df_temp_train.drop(to_drop, axis=1, inplace=True)# Drop features
    df_temp_test.drop(to_drop, axis=1, inplace=True)# Drop features
    X_train=df_temp_train.iloc[:,:].values
    X_test=df_temp_test.iloc[:,:].values
    return X_train,X_test
```

*בעקבות חקר נתונים לא ניכרה סיבת להוצאת OUTLIERS הוצאת שורות/עמודות ריקות, מילוי תאים ריקים וכד'

שלב 2 - המודלים

2.0 פיצול דאטה

כל הדאטה פוצל ל Train set ו Test set 80%-20%

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

ובנוסף בניתי פונקציה שמעריכה על המודל במספר פרמטרים שנלמדו.

```
def measuring_metrics(y_test,y_pred):
    mse=mean_squared_error(y_test,y_pred)
    rmse=np.sqrt(mse)
    mae=mean_absolute_error(y_test,y_pred)
    r2=r2_score(y_test,y_pred)
    print("mse:" + '\t', mse)
    print("rmse:" + '\t', rmse)
    print("mae:" + '\t', mae)
    print("r2:" + '\t', r2, '\n')
    metricss=[mse,rmse,mae,r2]
    return(metricss)
```

*KFOLD שנבדק בכל המודלים:

```
kfold = StratifiedKFold(n_splits=2, random_state=1, shuffle=True)
```

כל בנאי המודלים למעט רגרסיה לינארית נבנו לפני ההרצה והוכנסו לליסט יחד עם שם המודל ולאחר מכן בלולאה רצתי על כל המודלים כולל בדיקות הייפרמטרס ייחודיות.

```
models = []
models.append(('PR', LinearRegression(normalize=norma)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('LASSO', LassoCV(alphas=arange(0, 1, 0.01), cv=kfold)))
models.append(('RR', Ridge()))
models.append(('RF', RandomForestRegressor(random_state=1)))
```

```
for name, model in models:
```

לאחר כל ריצה אופטימלית על מודל הוכנס ערך הr2 עם שם המודל לליסט לצורך השוואה בסוף. (בתוך הפונקציה הוגדר הליסט כGlobal).

2.1 – Linear regression

שלב א – לאחר ניקוי הדאטה בשלב 1 הוספתי עמודה של 1 לפני הCONSTANT כפי שנלמד על מנת "לתת" משקל אליו.

```
X_train=np.append(arr=np.ones((508,1)).astype(int),values=X_train ,axis=1)
X_test=np.append(arr=np.ones((127,1)).astype(int),values=X_test ,axis=1)
```

שלב ב – ביצוע בחירת פיצ'ארים רלוונטיים לפי p-values באמצעות שיטת backward elimination. לולאה שמוריד כל פעם את הפיצ'ר בעל הp-values הגבוה ביותר עד שכל הפיצ'ארים בעלי p-values קטן מ0.05. כמו קודם הבדיקה התבצעה על הTRAIN SET אך התוצאה הושלכה גם על הTEST SET.

```
def backward_elimination(x,X_ols_test,y_dependent,SL):
    var=np.arange(x.shape[1])
    x_ols_array=x[:,var]
    regressor=sm.OLS(y_dependent,x_ols_array).fit()
    for i in range(sum(regressor.pvalues>0)):
        if sum(regressor.pvalues>SL)>0:
            arg=regressor.pvalues.argmax()
            var=np.delete(var,arg)
            x_ols_array=x[:,var]
            X_ols_test2=X_ols_test[:,var]
            regressor=sm.OLS(y_dependent,x_ols_array).fit()
    return (var[:],regressor,x_ols_array,X_ols_test2)
```

שלב ג – התבצעה הפרדיקציה לפי בנאי רגרסיה לינארית הודפסו תוצאות כמה מדדים נבחרים.

דוגמא לפי מספר חיוביים:

```
Linear Regresson metrics results:
mse:  66236.03115356683
rmse:     257.36361660803345
mae:  203.23036216917487
r2:   0.820860779484752
```

Polynomial regression - 2.2

שלב א – נבנה בנאי למודל רגרסייה לינארית שעליו יתבסס הרגרסייה הפולינומית.

```
models.append(('PR', LinearRegression(normalize=norma)))
```

שבה נכנס בדיקה האם כדאי לבצע נורמליזצייה על הנתונים בנוסף או לא.

```
norma = [True,False]
```

לאחר מכן התבצע בדיקה על המודל ונבדק על מעלה בגודל 1-5 מה הכי טוב והוחזר הפתרון הטוב ביותר עם המעלה המיטבית.

```
if (name=='PR'):
    degs = [1,2,3,4,5]
    best_score = 0
    best_deg = 0
    for deg in degs:
        for nor in norma:
            poly_features= PolynomialFeatures(degree = deg)
            X_train_poly = poly_features.fit_transform(X_train)
            model.fit(X_train_poly, y_train)
            scores = cross_val_score(model, X_train_poly, y_train,cv=kfold, scoring='r2')
            if max(scores) > best_score:
                best_score = max(scores)
                best_deg = deg
    print('Polyminal Regresson metrics results:')
    print("according to r2 in Polyminal Regresson the best degree is: ", best_deg)
    print("according to r2 in Polyminal Regresson the r2 best score is: ", best_score,'\n')
    r_results.append((name, best_score))
```

תוצאות הייפרמטרס עבור חיוביים: החזקה הטובה ביותר – 1.

תוצאות הייפרמטרס עבור שליליים: החזקה הטובה ביותר – 1.

Ridge regression - 2.3

נבנה בנאי ולאחר מכן התבצע חיפוש מיטבי לאלפא בין 0 ל0.1 בטווחים של 0.01 והוחלט על פי r2 על פיו נקבע האלפא האופטימלית ואומן המודל ולאחר מכן הוחזרו המדדים הנבדקים ותוצאת r2 הוכנסה לליסט לצורך השוואה סופית.

```
if (name=='RR'):
    grid = dict()
    grid['alpha'] = arange(0,0.1,0.01)
    # define search
    search = GridSearchCV(model, grid, scoring='r2', cv=kfold)
    # perform the search
    results = search.fit(X_train, y_train)
    y_pred = results.predict(X_test)
    print('{} is the Best alpha in Ridge Regresson according to {} '.format(results.best_params_['alpha'],
                                                                            'r2'.replace("_"," ")))

    measuring_metrics(y_test,y_pred)
    r_results.append((name, r2_score(y_test,y_pred)))
```

תוצאות הייפרמטרס עבור חיוביים: אלפא אופטימלית – 0.04 .

תוצאות הייפרמטרס עבור שליליים: אלפא אופטימלית – 0.05 .

Lasso regression - 2.4

נבדקו אלפא בטווח 0 עד 1 בטווח של 0.01.

```
models.append(('LASSO', LassoCV(alphas=arange(0, 1, 0.01), cv=kfold)))
```

כמו כן נבדק האלפא המיטבית, המודל אומן בהתאם והוחזרו ערכי המדידה של המודל.

```
if (name=='LASSO'):
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)
    print('LASSO regresson metrics results:')
    print('according to r2 best alpha in LASSO Regresson: %f' % model.alpha_)
    measuring_metrics(y_test,y_pred)
    r_results.append((name, r2_score(y_test,y_pred)))
```

תוצאות הייפרמטרס עבור חיוביים: אלפא אופטימלית – 0.16 .

תוצאות הייפרמטרס עבור שליליים: אלפא אופטימלית – 0.54 .

Random Forest - 2.5

נבדקו הפרמטרים המתבקשים – מספר העצים: 50-120 ברווח של 10 עצים.

ועומק העץ: 4-14 ברווח של 1.

המודל אומן בהתאם והוחזרו ערכי המדידה של המודל.

```
if (name=='RF'):
    param_grid = {'n_estimators': range(50,130,10),
                  'max_features': ['auto'],
                  'max_depth' : np.arange(4,15),
                  'criterion' :['mse']}
    search = GridSearchCV(estimator=model,param_grid=param_grid, scoring='r2', cv=kfold)
    results_rf = search.fit(X_train, y_train)
    y_pred = results_rf.predict(X_test)
    print('in RandomForest regresson {} is the Best depth according to {} '.format(search.best_params_['max_depth'],
                                                                                   'r2'.replace("_"," ")))
    print('in RandomForest regresson {} is the Best number of trees according to {} '.format(search.best_params_['n_estimators
                                                                                              'r2'.replace("_"," ")))
```

תוצאות הייפרמטרס עבור חיוביים: גודל עומק – 10, מספר עצים – 110.

תוצאות הייפרמטרס עבור שליליים: גודל עומק – 8, מספר עצים – 100.

KneighborsRegressor – 2.6

במודל זה כמתבקש לאחר הכנסת המשתנים למודל הוערכו מספר שכנים K אופטימלי לבדיקה והובחן באמצעות r2 הגדול ביותר.

בנוסף נבחן מדד p אשר בוחן האם עדיף למדוד את המרחקים על פי מרחק אוקלידי או מרחק מנהטן.

המודל אומן בהתאם והוחזרו ערכי המדידה של המודל.

תוצאות הייפרמטרס עבור חיוביים: מספר שכנים – 1 , מרחק מנהטן.

תוצאות הייפרמטרס עבור שליליים: מספר שכנים – 1, מרחק מנהטן.

```
if (name=='KNN'):
    p=[1,2] #p = 1, this is equivalent to using manhattan distance and p = 2 for euclidea
    n_neighbors = list(range(1,30))
    #Convert to dictionary
    hyperparameters = dict(n_neighbors=n_neighbors,p=p)
    #Use GridSearch
    clf = GridSearchCV(model, hyperparameters, cv=kfold, scoring = 'r2')
    #Fit the model
    best_model = clf.fit(X_train,y_train)
    #Print The value of best Hyperparameters
    print('KNN metrics results:')
    print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
    print('Best p:', best_model.best_estimator_.get_params()['p'])
    y_pred = clf.predict(X_test)
    r_results.append((name, r2_score(y_test,y_pred)))
```

<u>שלב 3 – תוצאות</u>

בקוד הוצא כפלט לכל מודל מה הHYPERPARAMETERS המיטביים ותוצאות מדדים mse, rmse, mae, r2 score . לצורך העניין אציג את תוצאות r2 של כל המודלים.

Inde	Type	Size	Value
0	tuple	2	('LR', 0.820860779484752)
1	tuple	2	('PR', 0.7405994059640846)
2	tuple	2	('KNN', 0.8531384064131471)
3	tuple	2	('LASSO', 0.8214686871414879)
4	tuple	2	('RR', 0.8188887135630089)
5	tuple	2	('RF', 0.8974618647465196)
6	tuple	2	('LR', 0.998331111230837)
7	tuple	2	('PR', 0.9964751132348307)
8	tuple	2	('KNN', 0.9913528630948689)
9	tuple	2	('LASSO', 0.9982908264496021)
10	tuple	2	('RR', 0.9982599964930192)
11	tuple	2	('RF', 0.9923923414064746)

תוצאות המודל לחיזוי מספר החיוביים

תוצאות המודל לחיזוי מספר השליליים

לצורך בחירת המודל חיזוי הטוב יותר עבור חיזוי שליליים וחיוביים נבחן באמצעות r2 score כך:

```
i=0
best_score_pos = 0
best_score_neg = 0
for result in r_results:
    if i<6:
        if result[1] > best_score_pos:
            best_score_pos = result[1]
            best_reg_name_pos = result[0]
    else:
        if result[1] > best_score_neg:
            best_score_neg = result[1]
            best_reg_name_neg = result[0]
    i+=1
```

והתוצאות:

```
Accroding to r2 in the positive section the best model is  RF
 and his r2 score is:  0.8974618647465196

Accroding to r2 in the negative section the best model is  LR
 and his r2 score is:  0.998331111230837
```

<u>שלב 4 – מסקנות</u>

שלב ניקוי הדאטה חשוב ואך קריטי אך קודם כל יש לשים לב כיצד הדאטה מתנהג.

במהלך המטלה נאלצתי לחקור את הנתונים, לפלח את חשיבות הימים בגרפים ועוד. על מנת להבין כיצד כדאי לי לנקות את הדאטה.

למשל התפלאתי למצוא כיצד בגלל התנהגות הדאטה כלל החיזויים בתחום השלילי יצאו עם r2 גבוה.

לאחר מכן בעת אימון המודלים וחזייה בתוצאות הבנתי את החשיבות לHyperParameters השונים וכיצד הטווחים של החישוב שלהם צריכים להתבצע על מנת להקל על המעבד בזמן ריצה ולהגיע לתוצאות טובות יותר.

ובאילו מדדי הערכה כדאי להשתמש בחישובים השונים במודלים.