Εξόρυξη Δεδομένων και Αλγόριθμοι Μάθησης

Εργαστηριακή Άσκηση 2021

Ον/μο: Γεώργιος Σκόνδρας

Τμήμα: Μηχανικών Η/Υ και Πληροφορικής

Α.Μ: 1020408

Περιεχόμενα

Περιβάλλον Υλοποίησης και Βιβλιοθήκες

Η εργασία Υλοποιήθηκε εξ' ολοκλήρου στη γλώσσα Python (έκδοση 3.7.10)

Για την υλοποίηση χρησιμοποιήθηκε το Περιβάλλον Google Colaboratory. Πρόκειται για μια cloud υπηρεσία, η οποία παρέχει ένα Jupyter Notebook περιβάλλον που μπορεί ο χρήστης να δημιουργήσει και να τρέξει κώδικα Python. Η επιλογή αυτή έγινε καθώς η υπηρεσία αυτή υποστηρίζει τις βασικές βιβλιοθήκες Μηχανικής Μάθησης τις οποίες έχει ήδη εγκατεστημένες. Επιπλέον παρέχει πρόσβαση σε αρκετά ισχυρά μηχανήματα με GPUs και TPUs που επιτυγχάνουν πολύ καλές επιδόσεις σε απαιτητικές εφαρμογές Μηχανικής Μάθησης.

Παρακάτω φαίνεται ο Πίνακας με τις εκδόσεις των εργαλείων/βιβλιοθηκών.

Language/Library	**Version**
Python	3.7.10
Pandas	1.1.5
Numpy	1.19.5
Matplotlib	3.2.2
Seaborn	0.11.1
Sklearn	0.22.2.post1
Tensorflow	2.4.1
Keras	2.4.3
Gensim	3.6.0

Εργασία 1: Πρόβλεψη Επιρρέπειας Ασθενών σε Εγκεφαλικό (Stroke Prediction Dataset)

A: Ανάλυση και Γραφική Αναπαράσταση του Dataset

Ο κώδικας για αυτό το ερώτημα βρίσκεται στο αρχείο
DM_1_A_healthcare_data_presentation.ipynb

Με τη χρήση της βιβλιοθήκης Pandas φορτώνουμε στη μεταβλητή *df* το αρχείο `healthcare-dataset-stroke-data.csv` υπό τη μορφή dataframe και με την συνάρτηση *head()* της Pandas βλέπουμε στο *Figure 1* τις πρώτες εγγραφές του αρχείου που αφορούν στις πληροφορίες των ασθενών

```
[5]  df.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Figure 1

Βλέποντας την τιμή του property shape του dataframe συμπεραίνουμε ότι το dataset έχει 5110 εγγραφές και 12 χαρακτηριστικά για κάθε εγγραφή. Οι στήλες με τις πληροφορίες για τους ασθενείς είναι οι εξής: **id**, **gender**, **age**, **hypertension**, **heart_disease**, **ever_married** , **work_type**, **Residence_type**, **avg_glucose_level**, **bmi**, **smoking_status**, **stroke**.

Τυπώνουμε το property dtypes του dataframe μας και βλέπουμε στο *Figure 2* τον τύπο δεδομένων της κάθε στήλης

```
[6]  df.dtypes
```

```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

Figure 2

Παρατηρούμε ότι έχουμε στήλες με τιμές τύπου:

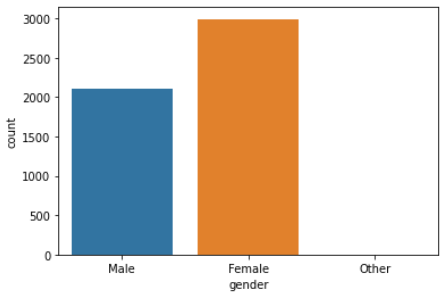
- **int64**(ακέραιες τιμές): **id**, **hypertension**, **heart_disease**, **stroke**
- **float64**(α.κ.υ.): **age**, **avg_glucose_level**, **bmi**
- **object***: **gender**, **ever_married**, **work_type**, **Residence_type**, **smoking_status**

*Σημείωση: Γενικά τύπος δεδομένων **object** μπορεί να περιέχει πολλούς διαφορετικούς τύπους(κείμενο ή μικτές αριθμητικές ή μη τιμές). Στην περίπτωση του dataset που μελετάμε, οι στήλες που περιέχουν δεδομένα τύπου object, αφορούν text data.

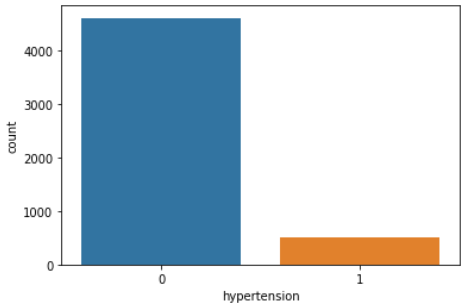
Σύμφωνα και με το αρχείο *Attribute Information.docx* που παρέχεται μαζί με το dataset παραθέτουμε τις πληροφορίες που μας παρέχουν οι τιμές των χαρακτηριστικών για τον κάθε ασθενή:

- **id**: μοναδικό αναγνωστικό για κάθε ασθενή
- **gender**: πληροφορία για το φύλο, "Male", "Female" ή "Other"
- **age**: ηλικία
- **hypertension**: 0 αν ο/η ασθενής δεν έχει υπέρταση, 1 αν έχει
- **heart_disease**: 0 αν ο/η ασθενής δεν έχει καρδιακό νόσημα, 1 αν έχει
- **ever_married**: πληροφορία για το αν έχει παντρευτεί ο/η ασθενής, "No" ή "Yes"
- **work_type**: πληροφορία για την εργασία του/της ασθενούς,"children", "Govt_jov", "Never_worked", "Private" ή "Self-employed"
- **Residence_type**: πληροφορία για τον τόπο κατοικίας του/της ασθενούς, "Rural" ή "Urban"
- **avg_glucose_level**: μέσο επίπεδο γλυκόζης στο αίμα
- **bmi**: δείκτης μάζας σώματος
- **smoking_status**: πληροφορία για το αν ο/η ασθενής καπνίζει ή κάπνιζε, "formerly smoked", "never smoked", "smokes" or "Unknown"(Η τιμή "Unknown" σημαίνει ότι η πληροφορία αυτή δεν είναι διαθέσιμή για αυτό τον ασθενή)
- **stroke**: 1 αν ο/η ασθενής είχε Εγκεφαλικό,0 αν δεν είχε

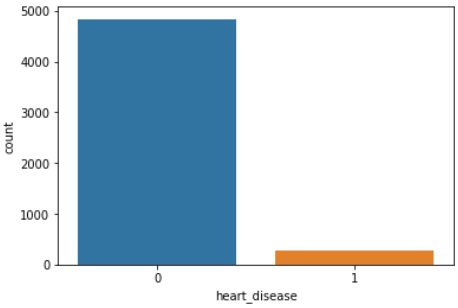
Όσον αφορά τα κατηγορικά και τα δεδομένα των οποίων οι τιμές είναι δυαδικές, δηλαδή τα χαρακτηριστικά: **gender**, **hypertension**, **heart_disease**, **ever_married**, **work_type, residence_type**, **smoking_status**, **stroke** βλέπουμε το πλήθος των εμφανίσεων της κάθε τιμής με τη χρήση της συνάρτησης *value_counts()* της **Pandas** και την αναπαριστούμε γραφικά με τη συνάρτηση *countplot()* της **Seaborn.** Να σημειωθεί πως κανένα από αυτά τα χαρακτηριστικά δεν έχει null τιμές.



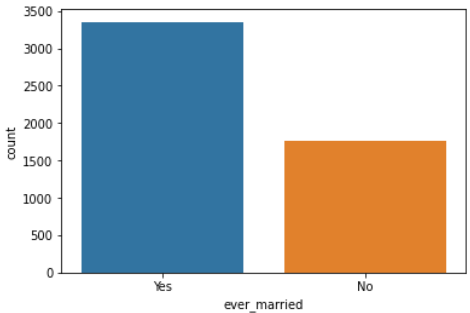
Gender: **Female**:2994, **Male**: 2115, **Other**: 1



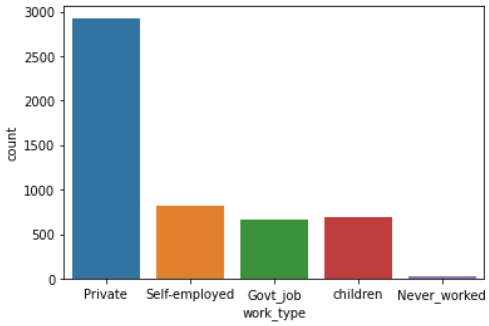
hypertension: **0**: 4612, **1**: 498



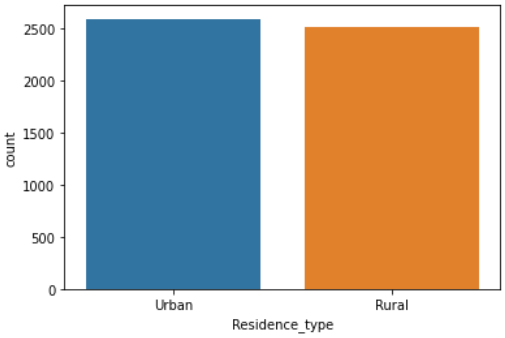
heart_disease: **0**: 4834, **1**: 276



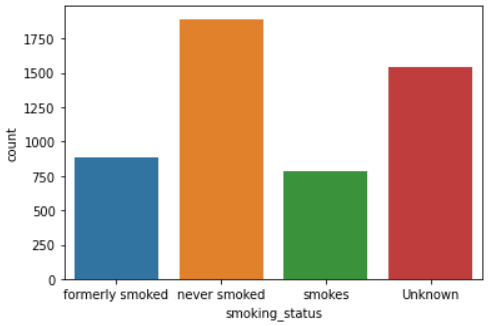
ever_married: **Yes**: 3335, **No**: 1757



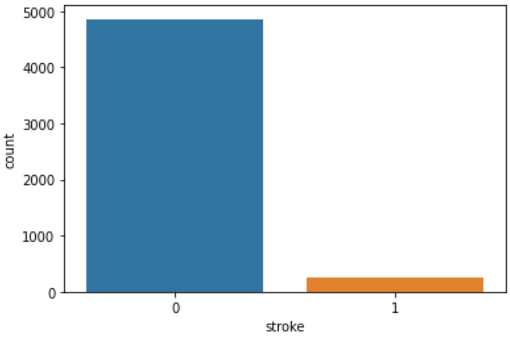
work_type: **Private**: 2925, **Self-employed**: 819, **children**: 687, **Govt_jov**: 657, **Never_worked**: 22



Residence_type: **Urban**: 2596, **Rural**: 2514



smoking_status: **never smoked**: 1892, **Unknown**: 1544, **formerly smoked**: 885, **smokes**: 789

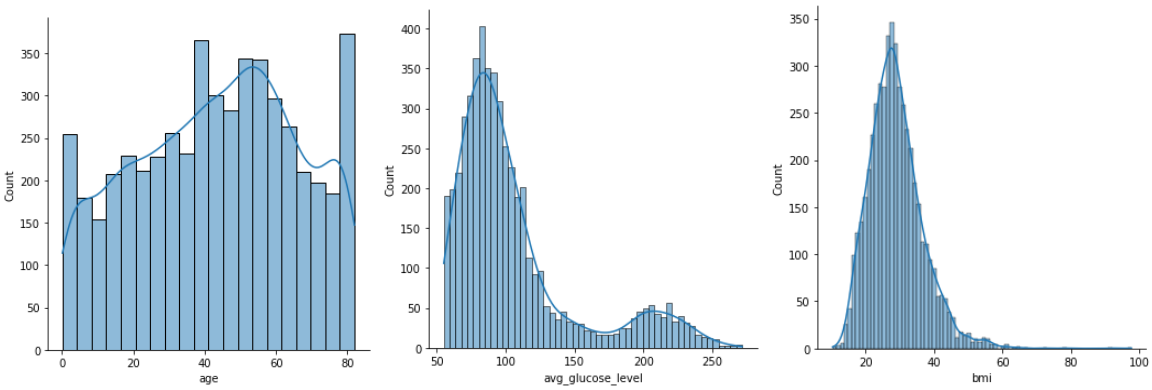


stroke: 0: 4861, **1**: 249

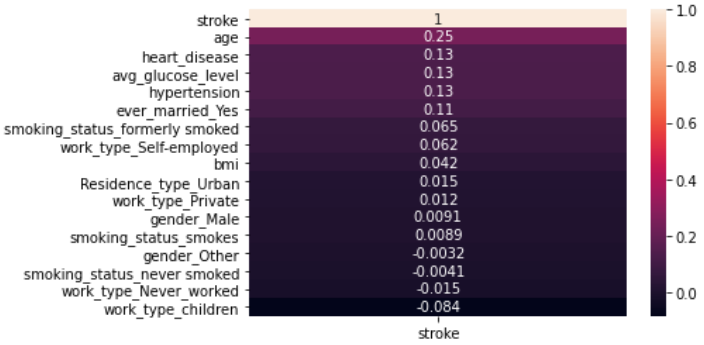
Όσον αφορά τα συνεχή αριθμητικά δεδομένα **age, avg_glucose_level, bmi** με τη βοήθεια των συναρτήσεων *describe()*, *median()*, *isnull()* προκύπτει ο πίνακας με τα παρακάτω στατιστικά στοιχεία που αφορούν στη μέγιστη, ελάχιστη τιμή, μέση τιμή, τυπική απόκλιση, διάμεσο και αριθμό null τιμών.

feature	Min	Max	Mean	Std	Median	#Null Values
age	0.08	82.00	43.22	22.61	45.00	0
avg_glucose_level	55.12	271.74	106.14	45.28	91.88	0
bmi	10.30	97.60	28.89	7.85	28.10	201

Με τη βοήθεια της συνάρτηση *displot()* της Seaborn βλέπουμε τις κατανομές των τιμών των χαρακτηριστικών **age, avg_glucose_level, bmi**

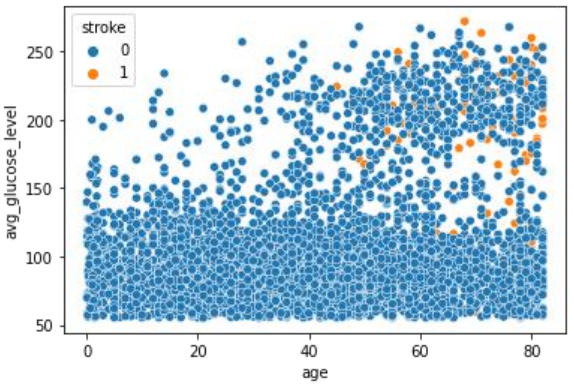


Στη συνέχεια θα μελετήσουμε τη συσχέτιση του χαρακτηριστικού **stroke** το οποίο θέλουμε να προβλέψουμε σε σχέση με τα υπόλοιπα χαρακτηριστικά. Κάνουμε χρήση της συνάρτησης *get_dummies()* της **Pandas** για τον χειρισμό των κατηγορικών δεδομένων και της συνάρτησης *corr()* για τον υπολογισμό των τιμών των **Συσχετίσεων Pearson**. Η επιλογή της μετρικής Pearson έγινε επειδή είναι από τις καλύτερες μεθόδους μέτρησης συσχετίσεων μεταξύ συνεχών μεταβλητών. Στη συνέχεια με τη συνάρτηση *heatmap()* της Seaborn αναπαριστούμε γραφικά τις συσχετίσεις του χαρακτηριστικού **stroke**



Η **Συσχέτιση** εκφράζει τη δύναμη και την κατεύθυνση της γραμμικής σχέσης που έχουν τα χαρακτηριστικά μεταξύ τους. Παίρνει τιμές στο [-1,1] όπου, 1 δείχνει πλήρη θετική συσχέτιση, -1 πλήρη αρνητική συσχέτιση και 0 καθόλου συσχέτιση. Στην προκειμένη περίπτωση βλέπουμε ότι τη μεγαλύτερη συσχέτιση με το **stroke** έχουν τα χαρακτηριστικά **age**, **heart_disease**, **avg_glucose_level**, **hypertension**. Δηλαδή η ύπαρξη εγκεφαλικού εξαρτάται αρκετά από την υψηλή ηλικία, την ύπαρξη καρδιακής νόσου και υπέρτασης καθώς και υψηλού ποσοστού γλυκόζης στο αίμα.

Στο παρακάτω scatterplot φαίνεται μία αναπαράσταση των δεδομένων που αφορά το χαρακτηριστικό **stroke** σε σχέση με τα **avg_glucose_level** και **age.** Πιο συγκεκριμένα οι κουκίδες με μπλε χρώμα αναπαριστούν τις εγγραφές που η τιμή του **stroke** είναι 0, ενώ οι πορτοκαλί αυτές που έχουν την τιμή 1. Στον οριζόντιο και στον κάθετο άξονα βρίσκονται οι αντίστοιχες τιμές των χαρακτηριστικών **age** και **avg_glucose_level** αντίστοιχα. Με αυτή τη γραφική επιβεβαιώνουμε τη μεγάλη συσχέτιση που δείξαμε προηγουμένως (με τη χρήση της Pearson Correlation) του χαρακτηριστικού **stroke** με τα χαρακτηριστικά **avg_glucose_level** και **age.** Όπως είναι εμφανές από τη γραφική οι πορτοκαλί κουκίδες(που αντιπροσωπεύουν τις εγγραφές με **stroke=1**) βρίσκονται κυρίως πάνω και δεξιά, δηλαδή εκεί που οι τιμές των **avg_glucose_level** και **age** είναι υψηλές.



B: Χειρισμός Ελλιπών Τιμών

Ο κώδικας για αυτό το ερώτημα βρίσκεται στο αρχείο *DM_1_B_missing_value_handling.ipynb*

Σε αυτό το ερώτημα θα εντοπίσουμε και θα χειριστούμε τις ελλιπείς τιμές. Όπως αναλύσαμε στο Ερώτημα Β. Το χαρακτηριστικό **bmi** έχει 201 Null τιμές και το χαρακτηριστικό **smoking_status** έχει 1544 τιμές “Uknown”. Όσον αφορά τις τιμές “Uknown” **του smoking_status** θα τις αντικαταστήσουμε αρχικά με Null τιμές για διευκόλυνση. Τις ελλιπείς τιμές θα χειριστούμε με τις παρακάτω μεθόδους

1: Αφαίρεση στήλης

Με τη χρήση της εντολής *dropna()* της **Pandas** αφαιρούμε τις στήλες **bmi** και **smoking_status** που έχουν ελλιπείς τιμές. Παρακάτω βλέπουμε τις 5 πρώτες εγγραφές του dataframe *df1* που προκύπτει:

```
[6] #Remove Columns with missing values (bmi and smoking_status)
    df1 = df.dropna(axis=1)
    df1.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	1

2: Συμπλήρωση Ελλιπών Τιμών με Μέσο όρο στήλης

Σε αυτό το ερώτημα θα συμπληρώσουμε μόνο τις ελλιπείς τιμές της στήλης **bmi** καθώς μόνο για αυτήν έχει νόημα ο μέσος όρος, εφόσον το **bmi** παίρνει συνεχείς αριθμητικές τιμές.

Αρχικά υπολογίζουμε το μέσο όρο της στήλης **bmi** όπως φαίνεται στη διπλανή εικόνα με **mean(bmi) = 28.89**

```
[8] # Calculata bmi mean
    bmi_mean = df['bmi'].mean()
    bmi_mean
```

```
28.893236911794673
```

Στη συνέχεια αντικαθιστούμε τις ελλιπείς τιμές με τη μέση τιμή της στήλης **bmi** και τυπώνουμε τις 5 πρώτες εγγραφές του dataframe *df2* που προκύπτει:

```
[9] # Replace bmi missing values with column mean
    df2 = df.copy()
    df2.bmi = df2['bmi'].fillna(value= bmi_mean)
    df2.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.600000	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.893237	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

3: Συμπλήρωση Ελλιπών Τιμών με Linear Regression

Σε αυτό το ερώτημα θα κάνουμε χρήση ενός μοντέλου **Linear Regression** για να προβλέψουμε τις ελλιπείς τιμές του χαρακτηριστικού **bmi**.

Αρχικά για να είναι αμερόληπτη η πρόβλεψη διαγράφουμε τις στήλες που περιέχουν την πληροφορία **stroke** και επιπλέον διαγράφουμε και τη στήλη **smokig_status** η οποία περιέχει επίσης ελλιπείς τιμές. Αυτό υλοποιείται το με τη χρήση της συνάρτησης *drop()* της **Pandas**.

```
[10] #Create dataframe withnout smoking status, stroke columns
     df3_training = df.drop(columns=['smoking_status','stroke']).dropna(axis=0)
     df3_training.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0
5	56669	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0

Κωδικοποίηση

Εν συνεχεία θα πρέπει να βρούμε την κατάλληλη κωδικοποίηση για τα κατηγορικά δεδομένα μας ώστε να τα μετατρέψουμε σε αριθμητικά. Αυτό είναι απαραίτητο, καθώς το μοντέλο δέχεται σαν είσοδο μόνο αριθμητικές τιμές. Τα χαρακτηριστικά που μας ενδιαφέρουν να κωδικοποιήσουμε είναι τα εξής: **gender, ever_married, work_type, Residence_type.**

Μία πιθανή κωδικοποίηση που μπορούμε να χρησιμοποιήσουμε είναι η label encoding κωδικοποίηση. Δηλαδή να κάνουμε map κάθε κατηγορική τιμή σε μία αριθμητική τιμή. Π.χ. για τη στήλη **work_type** να κάνουμε map την τιμή Private σε 1, Self-employed σε 2 κ.ο.κ. Η κωδικοποίηση αυτή δεν είναι η κατάλληλη για την περίπτωσή μας, καθώς δημιουργεί μία διάταξη για τις τιμές μας, η οποία δεν έχει κάποια φυσική σημασία και μπορεί να οδηγήσει σε «λάθος συμπεράσματα» το μοντέλο μας.

Έτσι, εφόσον για όλα τα χαρακτηριστικά αυτά δεν έχει νόημα η διάταξη των τιμών, για να αποφύγουμε το παραπάνω πρόβλημα, θα κάνουμε χρήση one-hot encoding, δηλαδή τη δημιουργία νέω στηλών για κάθε τιμή του εκάστοτε χαρακτηριστικού, με την τιμή 1 αν αληθεύει και 0 αλλιώς. Αυτό πραγματοποιείται με τη χρήση της συνάρτησης *get_dummies()* της **Pandas**.

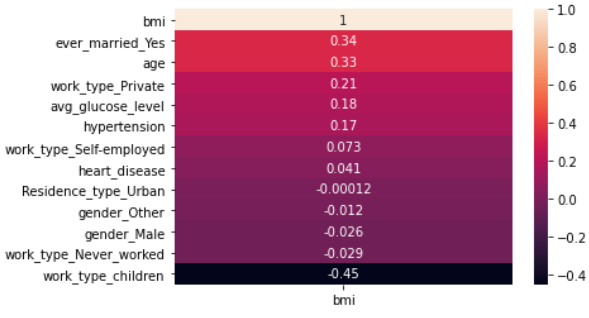
```
[12] #Get Dummy variables. Drop first column to avoid multicollinearity
     df3_training_dummies = pd.get_dummies(df3_training,drop_first=True).drop(['id'], axis=1)
     df3_training_dummies.head()
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	gender_Male	gender_Other	ever_married_Yes	work_type_Never_worked	work_type_Private	work_type_Self-employed	work_type_children	Residence_type_Urban
0	67.0	0	1	228.69	36.6	1	0	1	0	1	0	0	1
2	80.0	0	1	105.92	32.5	1	0	1	0	1	0	0	0
3	49.0	0	0	171.23	34.4	0	0	1	0	1	0	0	1
4	79.0	1	0	174.12	24.0	0	0	1	0	0	1	0	0
5	81.0	0	0	186.21	29.0	1	0	1	0	1	0	0	1

Επιπλέον στη συνάρτηση *get_dummies()* δίνουμε παράμετρο **drop_first= True** για να διαγραφεί η πρώτη στήλη που θα δημιουργηθεί για κάθε χαρακτηριστικό. Αυτό το κάνουμε για να αποφύγουμε το πρόβλημα του **multicollinearity,** δηλαδή να υπάρχουν συσχετίσεις μεταξύ ασυσχέτιστων χαρακτηριστικών. Στην περίπτωσή μας αποφεύγουμε το φαινόμενο αυτό, καθώς πλέον καμία στήλη δε μπορεί να προκύψει από τις υπόλοιπες του ίδιου αρχικού χαρακτηριστικού.

Μοντέλο

Στη συνέχεια θα βρούμε τα χαρακτηριστικά που έχουν τη μεγαλύτερη συσχέτιση με το **bmi** με τη χρήση της συνάρτησης *corr()* της Pandas και τις αναπαριστούμε γραφικά με τη χρήση της *heatmap()* της **Seaborn.** Παρατηρούμε ότι τη μεγαλύτερη συσχέτιση(κατ' απόλυτη τιμή) με το χαρακτηριστικό **bmi** έχουν τα εξής: **work_type_children, ever_married_yes, age, avg_glucose_level, hypertension.**



Για την εκτίμηση του μοντέλου, χωρίζουμε το υπάρχον dataset σε training και testing set(75%-25% split) με τη χρήση της *train_test_split()* της **Sklearn,** δημιουργούμε το μοντέλο Linear Regression με τη συνάρτηση *LinearRegression()* τη **Sklearn** με βάση το training set, προβλέπουμε τις υπόλοιπες τιμές και αξιολογούμε.

```
[22] # Train test split
     X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.25, random_state=42)
```

```
[23] #Fitting the test Linear Regression model based on 75% of the bmi data
     test_model = LinearRegression().fit(X_train, y_train)
```

```
[24] # Calculate predictions
     y_pred = test_model.predict (X_test)
```

```
[25] #Evaluating the test model
     print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
     print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

     Mean Absolute Error: 5.2118155586410175
     Root Mean Squared Error: 7.084010314801309
```

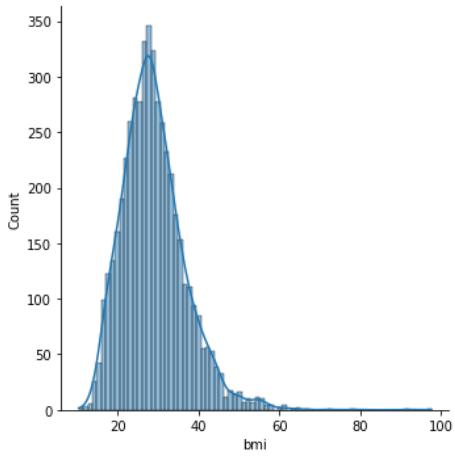
BMI Linear Regression Errors	MAE	RMSE
	5.21	7.08

Για να εκτιμήσουμε τις μετρικές σφάλματος MAE και RMSE, είναι χρήσιμο να βρούμε το εύρος των τιμών του **bmi,** που είναι 10.3 – 97.6 και να παρατηρήσουμε την κατανομή των τιμών με τη χρήση της *displot()*

```
[26] print(f"Bmi range: {df.bmi.min()} - {df.bmi.max()}")

     Bmi range: 10.3 - 97.6
```

Λαμβάνοντας αυτά υπ' όψη συμπεραίνουμε ότι το αποτέλεσμα της γραμμικής παλινδρόμησης δεν είναι πολύ ακριβές, αλλά παρ' όλα αυτά είναι ικανοποιητικό, καθώς έχουμε **Μέσο Απόλυτο Σφάλμα**: 5.21 και **Ρίζα Μέσου Τετραγωνικού Σφάλματος**: 7.08



Καταληκτικά δημιουργούμε ένα νέο μοντέλο Linear Regression πλέον με βάση όλα τα δεδομένα και συμπληρώνουμε όλες τις ελλιπείς τιμές του **bmi** και προκύπτει το dataframe *df3*. Τέλος βεβαιωνόμαστε ότι δεν υπάρχει καμία Null τιμή στη στήλη **bmi**, όπως φαίνεται παρακάτω:

```
[39] #Fill the missing values if bmi with the predicted ones
     df3['bmi'].iloc[bmi_missing_indeces] = predicted_bmi
```

```
[40] df3.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.600000	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	32.224675	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	1

```
[41] #Ensure there are no null values in bmi column
     print(f"Number of null values in bmi column:{df3['bmi'].isnull().sum(axis = 0)}")

     Number of null values in bmi column:0
```

4: Συμπλήρωση Ελλιπών Τιμών με k-Nearest Neighbors

Κωδικοποίηση

Σε αυτό το ερώτημα θα κάνουμε χρήση ενός μοντέλου K-Nearest Neighbors για να προβλέψουμε τις ελλιπείς τιμές του χαρακτηριστικού **smoking_status**. Αρχικά πρέπει να βρούμε την κατάλληλη κωδικοποίηση για το χαρακτηριστικό αυτό. Οι τιμές που παίρνει το **smoking_status** όπως δείξαμε στο Ερώτημα 1Α είναι οι εξής: “never smoked”, “formerly smoked”, “smokes”. Θεωρούμε πως οι τιμές αυτές έχουν μία διάταξη όσον αφορά το πόσο καπνιστής είναι κάποιος δηλαδή:

“smokes” > “formerly smoked” > “never smoked”

Η διάταξη αυτή μπορεί να είναι χρήσιμη για το μοντέλο μας και για να την εκμεταλλευτούμε θα κάνουμε χρήση label encoding. Πιο συγκεκριμένα θα κάνουμε το παρακάτω mapping.

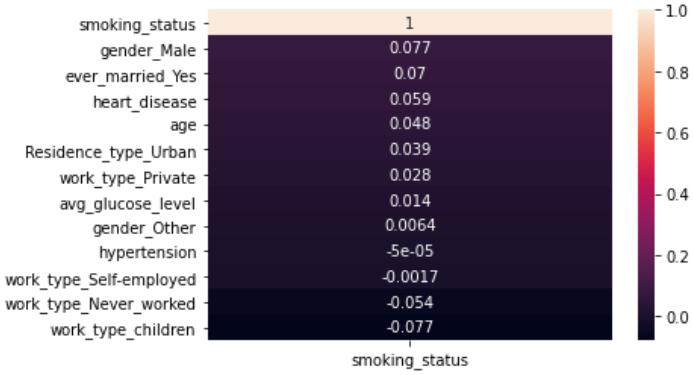
Smoking Status initial Value	Smoking Status Label Encoded Value
“never smoked”	0
“formerly smoked”	1
“smokes”	2

Για τα υπόλοιπα κατηγορικά χαρακτηριστικά, για τα οποία όπως αναφέραμε και στο προηγούμενο ερώτημα δεν έχει νόημα η διάταξη των τιμών τους θα κάνουμε χρήση one-hot encoding με τη χρήση της συνάρτησης *get_dummies()* όπως φαίνεται παρακάτω:

```
[46] #Get the one hot encoded dataset
     df4_training_dummies = pd.get_dummies(df4_training.drop(labels=['smoking_status'], axis=1), drop_first=True)
     df4_training_dummies['smoking_status'] = df4_training['smoking_status']
     df4_training_dummies.head()
```

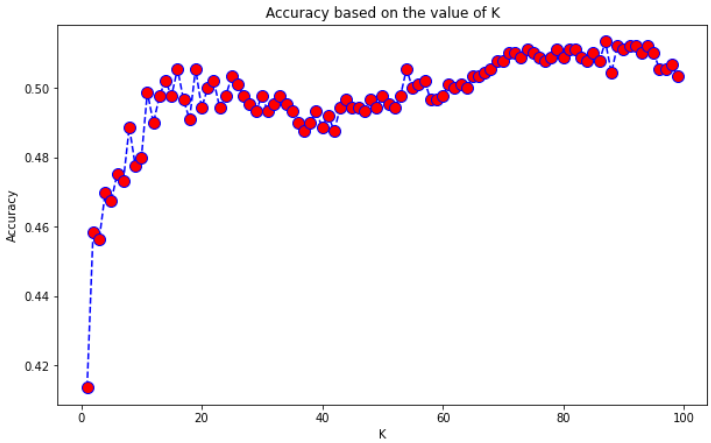
	age	hypertension	heart_disease	avg_glucose_level	gender_Male	gender_Other	ever_married_Yes	work_type_Never_worked	work_type_Private	work_type_Self-employed	work_type_children	Residence_type_Urban	smoking_status
0	67.0	0	1	228.69	1	0	1	0	1	0	0	1	1
1	61.0	0	0	202.21	0	0	1	0	0	1	0	0	0
2	80.0	0	1	105.92	1	0	1	0	1	0	0	0	0
3	49.0	0	0	171.23	0	0	1	0	1	0	0	1	2
4	79.0	1	0	174.12	0	0	1	0	0	1	0	0	0

Μοντέλο



Στη συνέχεια όμοια με τα προηγούμενα ερωτήματα θα υπολογίσουμε τις συσχετίσεις που έχουν όλα τα χαρακτηριστικά με το χαρακτηριστικό **smoking_status** ώστε να αποφασίσουμε ποια από αυτά θα χρησιμοποιήσουμε για την πρόβλεψή του. Και πάλι θα κάνουμε χρήση της συνάρτησης *corr()* της **Pandas** για των υπολογισμό των συσχετίσεων Pearson και της συνάρτησης *heatmap()* της **Seaborn** για την οπτικοποίηση. Παρατηρούμε ότι όλες οι τιμές των συσχετίσεων είναι κοντά στο 0 οπότε κανένα από τα χαρακτηριστικά φαίνεται να έχουν πολύ μικρή συσχέτιση με το χαρακτηριστικό **smoking_status**. Για το λόγο αυτό θα επιλέξουμε όλα τα χαρακτηριστικά για την πρόβλεψή του. Επιπλέον θα χωρίσουμε τα δεδομένα μας σε training και testing για την αξιολόγηση του μοντέλου μας

Επιλογή Κατάλληλου Κ



Για την επιλογή του κατάλληλου k θα εκπαιδεύσουμε 100 διαφορετικά μοντέλα με τα k να παίρνουν ακέραιες τιμές στο [1,99]. Υπολογίζουμε το accuracy για κάθε τιμή του k και βρίσκουμε ότι το καλύτερο k είναι το **k=86** που πετυχαίνει **accuracy = 51%.** Στην εικόνα φαίνεται και η καμπύλη με το accuracy για κάθε τιμή του k. Η συγκεκριμένη ακρίβεια δεν είναι αρκετά καλή. Αυτό οφείλεται στο γεγονός πως κανένα από τα χαρακτηριστικά δε σχετίζεται αρκετά με το χαρακτηριστικό **smoking_status** όπως προείπαμε. Τουλάχιστον η ακρίβεια **51%** που πετύχαμε είναι καλύτερη από μια τετριμμένη τυχαία επιλογή που θα είχε θεωρητικά ακρίβεια **33.33%.**

Στη συνέχεια δημιουργούμε ένα μοντέλο k-Nearest Neighbors με τη συνάρτηση *KNeighborsClassifier()*. Κάνουμε χρήση του μοντέλου αυτού και προβλέπουμε τις τιμές του **smoking_status** για τις ελλιπείς τιμές και στη συνέχεια αποκωδικοποιούμε τις αριθμητικές τιμές στις αντίστοιχες κατηγορικές και προκύπτει το dataframe *df4.* Επιπλέον βεβαιωνόμαστε ότι δεν υπάρχει καμία null τιμή στη στήλη **smoking_status**

```
[63] # Decode encoded smoking status labels
     smoking_decoded = df4['smoking_status'].str.replace('0','never smoked')
     smoking_decoded = smoking_decoded.str.replace('1',' formerly smoked')
     smoking_decoded = smoking_decoded.str.replace('2',' smokes')
     df4['smoking_status'] = smoking_decoded
     df4.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	never smoked	1

```
[64] #Ensure there are no null values in bmi column
     print(f"Number of null values in bmi column: {df4['smoking_status'].isnull().sum(axis = 0)}")

     Number of null values in bmi column: 0
```

Τέλος δημιουργούμε το dataset *df5* που περιέχει τις ελλιπείς τιμές σε όλα τα χαρακτηριστικά συμπληρωμένες. Πιο συγκεκριμένα οι ελλιπείς τιμές της στήλης **bmi** έχουν συμπληρωθεί σύμφωνα με το μοντέλο Linear Regression του Ερωτήματος Β3, ενώ αυτές της στήλης **smoking_status** με το μοντέλο k-Nearest Neighbors του Ερωτήματος Β4.

```
[65] df5 = df.copy()
     df5['bmi'] = df3['bmi']
     df5['smoking_status'] = df4['smoking_status']
     df5.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.600000	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	32.224675	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.500000	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.400000	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.000000	never smoked	1

Γ: Πρόβλεψη Επιρρέπειας Ασθενών σε Εγκεφαλικό με Μοντέλο Random Forest

Ο κώδικας για αυτό το ερώτημα βρίσκεται στο αρχείο *DM_1_C_stroke_predictions.ipynb*

Σε αυτό το σημείο θα εκπαιδεύσουμε ένα μοντέλο Random Forest προβλέψουμε αν ένας ασθενής είναι επιρρεπής ή όχι να πάθει εγκεφαλικό.

Όσον αφορά τα δεδομένα εκπαίδευσης, μετά τον χειρισμό των ελλιπών τιμών στο **Ερώτημα 1Β** προέκυψαν 5 dataset. Έτσι για κάθε μοντέλο θα γίνει εκπαίδευση και με τα 5 dataset τα οποία θα τοποθετηθούν σε κατάλληλες λίστες για το κάθε μοντέλο και θα κάνουμε διαχωρισμό training set-test set 75%-25%

Αρχικά εκπαιδεύουμε ένα μοντέλο Random Forest με τη χρήση της βιβλιοθήκης **Sklearn** με τις default παραμέτρους.

Παρακάτω αναλύονται και οι μετρικές ακρίβειας του καλύτερου μοντέλου ως τώρα, οι οποίες υπολογίστηκαν με τη συνάρτηση *classification_report()* του module metrics της **Sklearn**.

Όσον αφορά τη συμπεριφορά του μοντέλου στο training set βλέπουμε για όλα τα training dataset πετύχαμε accuracy 100%. Βέβαια εμάς μας ενδιαφέρει πως συμπεριφέρεται το μοντέλο στο test set. Εκεί το καλύτερο αποτέλεσμα επιτυγχάνεται με το 4^{ο} dataset(συμπλήρωση των τιμών του χαρακτηριστικού **smoking-status** με την τεχνική **K-Nearest Neighbors**) πετυχαίνοντας

Accuracy: 95%

Βλέπουμε ότι και με το πρώτο κιόλας μοντέλο η ακρίβεια ήταν αρκετά υψηλή. Δυστυχώς όμως η τιμή του Accuracy 95% είναι παραπλανητική. Όπως είδαμε στο **Ερώτημα 1Α** το dataset που έχουμε παρουσιάζει ανισορροπία όσον αφορά το χαρακτηριστικό **stroke.** Πιο συγκεκριμένα από τις 5110 εγγραφές, μόνο οι 249 έχουν **stroke=1**, ενώ όλες οι υπόλοιπες 4861 που αντιστοιχούν περίπου στο 95% του dataset έχουν **stroke=0.** Αυτό σημαίνει πως απλά επιλέγοντας για όλες τις εγγραφές εισόδου να έχουν την τιμή **stoke=0** ανεξάρτητα με τις τιμές των υπολοίπων χαρακτηριστικών θα πετυχαίναμε τιμές του **Accuracy** κοντά στο **95%!!!** Συνεπώς η ακρίβεια που πετύχαμε δεν είναι καλή.

Ας δούμε τώρα και τις υπόλοιπες μετρικές για να εξηγήσουμε τα αποτελέσματα του μοντέλου μας.

Πρώτα να εξηγήσουμε τι αναπαριστούν οι μετρικές **Accuracy**, **Precision, Recall, f1-score**

Για το confusion matrix του διπλανού πίνακα για ένα πρόβλημα δυαδική κατηγοριοποίησης.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Ορίζουμε τις μετρικές:

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

Accuracy: Μετρική που μας δείχνει πόσο καλά αποδίδει η δυαδική κατηγοριοποίηση. Προκύπτει από το πηλίκο των σωστών προβλέψεων προς το σύνολο των περιπτώσεων.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Precision: Μετρική που μας δείχνει από όλες τις θετικές κλάσεις που προβλέψαμε, πόσες ήταν όντως θετικές.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

Recall: Μετρική που μας δείχνει από όλες τις θετικές κλάσεις, πόσες από αυτές προβλέψαμε σωστά.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

F-1 score: Μετρική που συμπεριλαμβάνει το Precision και το Recall

Παρακάτω βλέπουμε τις μέσες τιμές των μετρικών **Precision, Recall, f1-score**

Overall Metrics	**Precision**	**Recall**	**f1-score**
Macro Average	0.56	0.51	0.50
Weighted Average	0.91	0.95	0.92

Να σημειωθεί πως οι Weighted Average μετρικές λαμβάνουν υπ' όψιν την ανισορροπία που έχει το dataset, σε σχέση με τις Macro Average μετρικές.

Παρακάτω βλέπουμε τις αντίστοιχες τιμές των μετρικών για κάθε κλάση ξεχωριστά

Metrics Per Class	**Precision**	**Recall**	**f1-score**
No-Stroke	0.95	1.00	0.97
Stroke	0.17	0.02	0.03

Είναι φανερό πως το μοντέλο μας έχει κάνει **overfit**, δηλαδή έχει υπερ-προσαρμοστεί στα δεδομένα του training set. Όλες οι μετρικές στο training set είχαν την τιμή 1, δηλαδή το μοντέλο έκανε τέλεια κατηγοριοποίηση. Στο test set το μοντέλο έχει μέτρια απόδοση. Πιο συγκεκριμένα έχουμε σχεδόν μηδενική ακρίβεια πρόβλεψης **stroke=1**, ενώ το μοντέλο έχει υπερ-προσαρμοστεί μόνο στο να προβλέπει καλά τις εγγραφές με **stroke=0.**

Το γεγονός αυτό οφείλεται στην **Ανισορροπία του Συνόλου Δεδομένων για το Χαρακτηριστικό stroke.** Μια συνηθισμένη τεχνική για να αντιμετωπίσουμε τέτοιες ανισορροπίες είναι η **Υπερδειγματοληψία** του dataset. Με την τεχνική αυτή δημιουργούμε τυχαία δείγματα επαναλαμβάνοντας τα ήδη υπάρχοντα. Κάνουμε τυχαία υπερδειγματοληψία με τη χρήση της συνάρτησης *RandomOverSampler()* της **Sklearn** και εκπαιδεύουμε και πάλι το μοντέλο με τις default παραμέτρους.

Το καλύτερο αποτέλεσμα το είχαμε και πάλι για το 4^{ο} dataset. Παρατηρούμε ότι η ακρίβεια μειώθηκε σε ποσοστό 1% και πλέον έχουμε **Accuracy 94%**, όμως αυξήθηκαν οι τιμές των μετρικών που έχουν σχέση με την πρόβλεψη του **stroke=1** όπως βλέπουμε παρακάτω.

Overall Metrics	**Precision**	**Recall**	**f1-score**
Macro Average	0.60	0.53	0.54
Weighted Average	0.92	0.94	0.93

Metrics Per Class	**Precision**	**Recall**	**f1-score**
No-Stroke	0.95	0.99	0.97
Stroke	0.24	0.08	0.12

Στη συνέχεια θα κάνουμε χρήση των τεχνικών Υπερδειγματοληψίας **Smote** και **Adasyn** με τη χρήση των αντίστοιχων βιβλιοθηκών της **Sklearn** αλλά τα αποτελέσματα δεν έχουν μεγάλη διαφορά από αυτά της **Τυχαίας Υπερδειγματοληψίας.**

Τέλος θα προσπαθήσουμε να βρούμε τις καλύτερες υπερ-παραμέτρους για το μοντέλο **Random Forest.** Μία καλή τεχνική για την εύρεση των βέλτιστων υπερ-παραμέτρων είναι το **GridSearch.** Με την τεχνική αυτή ορίζουμε ένα πλέγμα με τις τιμές των υπερ-παραμέτρων του μοντέλου σε κάποιο εύρος. Στη συνέχεια εκπαιδεύουμε για κάθε συνδυασμό των υπερ-παραμέτρων και εκτιμούμε τον καλύτερο συνδυασμό. Εμείς θα κάνουμε χρήση των παρακάτω τιμών των παραμέτρων, ορίζοντας σαν scoring μετρική τη μετρική recall στην οποία έχουμε τα χαμηλότερα αποτελέσματα.

```
parameters = {
    'n_estimators': (10,25,50,75,100),
    'criterion':('gini', 'entropy'),
    'max_depth': (3,7,18),
    'max_features':('auto','sqrt', 'log2'),
    'min_samples_split':(2,4,6),
    'class_weight': ('balanced', 'balanced_subsample', None)
    }
```

Η επιλογή του recall σαν scoring μετρική έγινε επειδή στην κατηγοριοποίηση της κλάσης **stroke=1**, λόγω της φύσης του προβλήματος θέλουμε να ανιχνεύσουμε όσο περισσότερους ασθενείς που έχουν επιρρέπεια σε εγκεφαλικό. Έχουμε μαγάλο κόστος αν κάποιος ασθενής με επιρρέπεια δεν ανιχνευτεί. Λόγω της φύσης της μετρικής recall προσπαθούμε έτσι από τους ασθενείς που έχουν επιρρέπεια να βρούμε τους περισσότερους.

Με την επιλογή των παραπάνω παραμέτρων οδηγούμαστε σε 5*2*3*3*3*3= **810 συνδυασμούς υπερ-παραμέτρων**.

Παρακάτω βλέπουμε τις μετρικές για το καλύτερο μοντέλο που προέκυψε

Accuracy: 95%

Overall Metrics	**Precision**	**Recall**	**f1-score**
Macro Average	0.63	0.53	0.54
Weighted Average	0.92	0.95	0.93

Metrics Per Class	**Precision**	**Recall**	**f1-score**
No-Stroke	0.95	0.99	0.97
Stroke	0.31	0.06	0.01

Παρατηρούμε ότι με τη χρήση του **Gridsearch** πετύχαμε μικρή βελτίωση της τάξης του 1-2% στις περισσότερες μετρικές αξιολόγησης.

Εργασία 2: Ανίχνευση Ανεπιθύμητης Αλληλογραφίας (Spam or not Spam Dataset)

Ο κώδικας για αυτό το ερώτημα βρίσκεται στο αρχείο *DM_2_spam_or_not_spam.ipynb*

Προεπεξεργασία

Αρχικά φορτώνουμε το dataset σε μορφή Pandas Dataframe και διαπιστώνουμε πως περιέχει 2 στήλες. Η μία περιέχει το κείμενο από διάφορα emails και η δεύτερη έχει την πληροφορία αν είναι spam ή όχι. Στο dataset περιέχονται 1500 εγγραφές.

```
[3] df = pd.read_csv('drive/MyDrive/DataMining/spam_or_not_spam/spam_or_not_spam.csv')
```

```
[4] df
```

	email	label
0	mike bostock said received from trackingNUMBE...	0
1	no i was just a little confused because i m r...	0
2	this is just an semi educated guess if i m wro...	0
3	jm URL justin mason writes except for NUMBER t...	0
4	i just picked up razor sdk NUMBER NUMBER and N...	0
...	...	...
1495	abc s good morning america ranks it the NUMBE...	1
1496	hyperlink hyperlink hyperlink let mortgage le...	1
1497	thank you for shopping with us gifts for all ...	1
1498	the famous ebay marketing e course learn to s...	1
1499	hello this is chinese traditional 子 件 NUMBER世...	1

1500 rows × 2 columns

Προτού ξεκινήσουμε ελέγχουμε για ελλιπείς τιμές. Βλέπουμε ότι υπάρχει μία εγγραφή με NaN τιμή στο πεδίο **email**. Η εγγραφή αυτή στο πεδίο label έχει την τιμή 1. Θα μπορούσαμε να αντικαταστήσουμε τη NaN τιμή με κάποια άλλη όπως π.χ. τον κενό χαρακτήρα (“ “).

```
[5] #Check for missing values on email column
    df.isnull().sum()
```

```
email    1
label    0
dtype: int64
```

```
[6] #Print the record with the missing value
    df.loc[df.email.isnull()]
```

	email	label
1466	NaN	1

Ελέγχουμε αν υπάρχουν άλλες εγγραφές που έχουν τον κενό χαρακτήρα στο πεδίο **email** και βλέπουμε ότι υπάρχουν 2 τέτοιες εγγραφές και μάλιστα με την ίδια τιμή ένα στο πεδίο **label.** Έτσι αντικαθιστούμε τη NaN τιμή στο πεδίο email με τον κενό χαρακτήρα.

```
[7] # Check for other records with a space on email column
    df.loc[df.email == ' ']
```

	email	label
1306		1
1328		1

```
[8] # Replace the null value with a space
    df.fillna(value= ' ', inplace=True)
```

Word Embeddings

Εφόσον θα κάνουμε χρήση ενός μοντέλου **Νευρωνικού Δικτύου** για να «μαντέψουμε» την πληροφορία της δεύτερης στήλης, θα πρέπει πρώτα να μετατρέψουμε το κείμενο των emails σε κατάλληλη μορφή, συμβατή με το μοντέλο μας. Για να γίνει αυτό θα κάνουμε χρήση της τεχνικής **Word Embeddings** χρησιμοποιώντας τη βιβλιοθήκη **Gensim** και εκπαιδεύοντας το δικό μας μοντέλο με τη συνάρτηση *Doc2Vec()*.

Μετατρέπουμε τα δεδομένα της στήλης **email** σε μία λίστα *email_docs* που κάθε στοιχείο της περιέχει ένα αντικείμενο της κλάσης *TaggedDocument()*. Δηλαδή μια λίστα με όλες της λέξεις που περιέχει η εκάστοτε εγγραφή και ένα μοναδικό αναγνωριστικό όπως φαίνεται παρακάτω.

```
[10] # Tokenize and tag the email text
     email_docs = [TaggedDocument(doc.split(' '), [i]) for i, doc in enumerate(df.email)]
```

```
[11] # Display the first 10 tagged docs from the list
     email_docs[:10]
```

```
[TaggedDocument(words=['', 'mike', 'bostock', 'said', 'received', 'from', 'trackingNUMBER', 'URL', 'NUMBER', 'NUMBER', 'NUMBER', 'NUMBER', 'by', 'URL', 'postfix', 'with', 'esmtp', 'id', 'NUMBERedNUMBER', 'for',
 TaggedDocument(words=['', 'no', 'i', 'was', 'just', 'a', 'little', 'confused', 'because', 'i', 'm', 'running', 'procmail', 'on', 'a', 'gateway', 'and', 'sits', 'between', 'the', 'external', 'sendmail', 'box', '
 TaggedDocument(words=['this', 'is', 'just', 'an', 'semi', 'educated', 'guess', 'if', 'i', 'm', 'wrong', 'someone', 'please', 'correct', 'me', 'spamd', 'setuid', 's', 'to', 'the', 'user', 'running', 'spamc', 'si
 TaggedDocument(words=['jm', 'URL', 'justin', 'mason', 'writes', 'except', 'for', 'NUMBER', 'thing', 'defanged', 'mime', 'messages', 'that', 's', 'a', 'big', 'problem', 'but', 'if', 'you', 'didn', 't', 'just', '
 TaggedDocument(words=['i', 'just', 'picked', 'up', 'razor', 'sdk', 'NUMBER', 'NUMBER', 'and', 'NUMBER', 'NUMBER', 'agents', 'from', 'the', 'the', 'razor', 'site', 'i', 'am', 'using', 'suse', 'NUMBER', 'NUMBER',
 TaggedDocument(words=['on', 'sep', 'NUMBER', 'NUMBER', 'NUMBERpm', 'daniel', 'quinlan', 'wrote', 'allen', 'smith', 'easmith', 'beatrice', 'rutgers', 'edu', 'writes', 'well', 'i', 'have', 'been', 'doing', 'a', '
 TaggedDocument(words=['unable', 'to', 'find', 'user', 'matt_relay', 'sbcglobal', 'net', 'please', 'make', 'sure', 'the', 'address', 'is', 'correct', 'and', 'resend', 'your', 'mail', ''], tags=[6]),
 TaggedDocument(words=['', 'spamtalk', 'said', 'probably', 'better', 'than', 'the', 'spam', 'phrases', 'approach', 'would', 'be', 'the', 'database', 'approach', 'as', 'currently', 'used', 'for', 'white', 'black'
 TaggedDocument(words=['on', 'sun', 'sep', 'NUMBER', 'NUMBER', 'at', 'NUMBER', 'NUMBER', 'NUMBERpm', 'NUMBER', 'justin', 'mason', 'wrote', 'i', 'seem', 'to', 'be', 'getting', 'a', 'lot', 'of', 'spam', 'relayed',
 TaggedDocument(words=['jm', 'URL', 'justin', 'mason', 'writes', 'yes', 'NUMBER', 'of', 'the', 'entire', 'set', 'for', 'training', 'and', 'NUMBER', 'for', 'evaluation', 'once', 'you', 've', 'settled', 'on', 'the
```

Στη συνέχεια αρχικοποιούμε ένα μοντέλο *Doc2Vec()* δίνοντας του σαν παράμετρο **vector_size=64** ώστε το κάθε επιμέρους κείμενο να αντιστοιχιστεί σε ένα διάνυσμα μεγέθους 64 και σαν αριθμό εποχών εκπαίδευσης **epochs=40**. Δημιουργούμε ένα λεξιλόγιο και το εκπαιδεύουμε με βάση τα κείμενα *email_docs* που δημιουργήσαμε προηγουμένως.

```
[12] # Instantiate model
     doc_model = Doc2Vec(vector_size=64, window=2, min_count=1, workers=8, epochs = 40)
```

```
[13] # Build vocab
     doc_model.build_vocab(email_docs)
```

```
[14] # Train model
     doc_model.train(email_docs, total_examples=doc_model.corpus_count, epochs=doc_model.epochs)
```

Στη συνέχεια με τη χρήση της συνάρτησης *infer_vector()* για το μοντέλο που δημιουργήσαμε, κατασκευάζουμε τα αριθμητικά διανύσματα που αντιστοιχούν στο κείμενο των email. Παρακάτω φαίνονται τα δύο πρώτα διανύσματα.

```
[16] # Generate vectors
     email_vectors = [doc_model.infer_vector((df['email'][i].split(' '))) for i in range(0,len(df['email']))]
     email_vectors[:2]
```

```
[array([ 8.62066984e-01, -1.13814104e+00,  3.23044300e-01, -8.40051472e-01,
         1.48061061e+00, -4.13247049e-01,  1.14724815e-01,  8.29123199e-01,
         3.06187570e-01,  1.38397062e+00, -5.55120170e-01,  4.01679128e-01,
        -1.42961705e+00,  4.63483073e-02,  1.78677756e-02,  3.23023349e-01,
        -5.47126532e-01, -2.76048005e-01,  2.40759969e-01, -7.68795788e-01,
        -1.41466439e+00, -4.26852435e-01,  3.10889244e-01,  1.36862838e+00,
         7.36609757e-01, -1.80011010e+00,  1.05790520e+00,  1.78336644e+00,
        -4.11855787e-01, -8.91842097e-02,  4.80385482e-01,  1.82662010e+00,
        -4.04510535e-02, -1.99335206e+00, -3.47206652e-01, -2.73825061e-02,
         1.37334001e+00,  2.88494416e-02,  2.55629182e-01, -1.58733952e+00,
        -8.50443184e-01, -3.33502501e-01, -4.62562323e-01,  7.96794370e-02,
        -1.70784783e+00, -6.75013542e-01,  5.07960737e-01, -1.22389925e+00,
         3.89040112e-01,  8.00446391e-01,  1.16353166e+00, -5.86776257e-01,
        -5.29256165e-01, -3.94402623e-01, -2.56577671e-01,  3.17637950e-01,
         4.06772655e-04, -8.62597674e-02,  7.29210675e-01,  7.02768028e-01,
         2.71451175e-01,  7.90241137e-02,  5.84113657e-01, -5.55893421e-01],
       dtype=float32),
 array([ 0.4565078 , -1.9327291 , -0.15174653,  1.4597713 , -0.5754755 ,
         0.43148202,  0.4106399 , -0.38998717,  2.827443  ,  1.7639476 ,
        -0.04929277, -0.06804072, -0.0552612 ,  0.9686009 , -0.7907614 ,
        -1.6210742 , -1.6355722 ,  0.13982157,  0.65898585, -2.165356  ,
         0.8412757 ,  0.15956888,  0.99588203,  1.1719398 ,  1.9651539 ,
        -1.0044332 ,  0.06665705,  1.5315342 ,  0.30492043, -0.06665502,
         3.1469135 ,  0.90788114, -0.97554815, -1.9231057 ,  0.32090282,
        -0.70505375, -0.394079  , -1.091531  , -0.53651655,  0.9923148 ,
         0.22334404, -1.3611735 ,  0.5074046 ,  1.8162907 , -1.4594812 ,
        -0.675555  , -1.9526825 , -1.829688  , -1.0443563 ,  0.92407507,
         1.0224051 , -0.7770362 ,  0.3046627 , -0.5593587 , -4.229377  ,
         0.01348625,  0.80154914,  0.9172394 ,  0.67368925, -1.2737219 ,
        -0.450514  ,  1.6400462 ,  2.1548002 ,  2.43849   ], dtype=float32)]
```

Μετατρέπουμε τα διανύσματα στην κατάλληλη μορφή και τα προσθέτουμε σαν επιπλέον στήλη του dataset.

```
[17] # Create a list of lists
     dtv= np.array(email_vectors).tolist()
     #set list to dataframe column
     df['email_vectors'] = dtv
     df.head()
```

	email	label	email_vectors
0	mike bostock said received from trackingNUMBE...	0	[0.8620669841766357, -1.1381410360336304, 0.32...
1	no i was just a little confused because i m r...	0	[0.4565078020095825, -1.9327291250228882, -0.1...
2	this is just an semi educated guess if i m wro...	0	[-0.04583593085408211, -1.2303012609481812, 0....
3	jm URL justin mason writes except for NUMBER t...	0	[0.40921279788017273, -0.9609760046005249, 0.3...
4	i just picked up razor sdk NUMBER NUMBER and N...	0	[0.36482328176498413, -1.5464613437652588, -0....

Χωρίζουμε το dataset σε training/testing με τη συνάρτηση *train_test_split()* της **Sklearn** με αναλογία 75%-25%**.**

Μοντέλα

Με τη βιβλιοθήκη **Keras** πάνω από **Tensorflow** ορίζουμε αρχικά ένα μοντέλο **Νευρωνικού Δικτύου** όπως φαίνεται παρακάτω:

```
[64] # Define keras model
     model_0 = Sequential()
     model_0.add(Dense(30, input_dim=vector_size, activation='relu'))
     model_0.add(Dense(1, activation='sigmoid'))
```

Θέτουμε:

- 1 κρυφό επίπεδο 30 Νευρώνων με συνάρτηση ενεργοποίησης Rectified Linear Unit(ReLU). Η επιλογή της ReLU έγινε λόγω της απλότητας και της ταχύτητας που προσφέρει ταυτόχρονα με τη μη γραμμικότητα που της επιτρέπει να διαδίδει αποτελεσματικά τα σφάλματα προς τα πίσω στα επίπεδα του Νευρωνικού Δικτύου. Ο αριθμός των Νευρώνων επιλέχθηκε πειραματικά.
- 1 Νευρώνα στο επίπεδο εξόδου με σιγμοειδή συνάρτηση ενεργοποίησης. Επιλέξαμε 1 Νευρώνα καθώς έχουμε πρόβλημα κατηγοριοποίησης και η τιμή 1 στην έξοδο αναπαριστά spam και η τιμή 0 μη-spam e-mail. Η sigmoid συνάρτηση ενεργοποίησης είναι κατάλληλη για τέτοια προβλήματα καθώς κάνει map πραγματικές τιμές στο διάστημα [0,1].

  Στη συνέχεια κάνουμε compile το μοντέλο όπως φαίνεται παρακάτω:

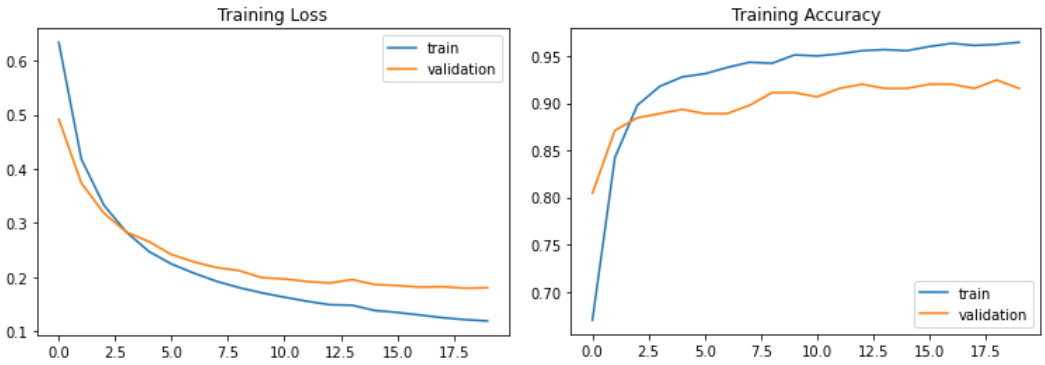
```
model_0.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history_0 = model_0.fit(X_train, y_train, epochs=20,
          validation_split=0.2,
          )
```

Θέτουμε:

- **Συνάρτηση σφάλματος** τη συνάρτηση **Binary Cross-Entropy**, η οποία είναι κατάλληλη για προβλήματα δυαδικής κατηγοριοποίησης, όπως είναι το δικό μας πρόβλημα.
- **Optimizer**, δηλαδή αλγόριθμο υπεύθυνο για την ενημέρωση των παραμέτρων του Νευρωνικού Δικτύου(βάρη, ρυθμό εκπαίδευσης) με σκοπό τη μείωση του σφάλματος τον **Adam** Optimizer

- Ως **μετρική αξιολόγησης** για το μοντέλο κατά τη διάρκεια του training και του testing, το **accuracy** του μοντέλου. Δηλαδή το πηλίκο των σωστών εκτιμήσεων του μοντέλου προς το σύνολο των εκτιμήσεων.
- **20 εποχές εκπαίδευσης**, αριθμός που προέκυψε πειραματικά.
- **Validation split** σε ποσοστό **80%-20%** ώστε να μπορούμε να παρατηρούμε κατά την εκπαίδευση του μοντέλου πως θα απέδιδε σε άγνωστα δεδομένα και να ελέγχουμε κατά πόσο το μοντέλο κάνει overfit στα δεδομένα εκπαίδευσης.

Εκπαιδεύουμε το μοντέλο και στις παρακάτω γραφικές φαίνεται η ακρίβεια και το σφάλμα του μοντέλου κατά την εκπαίδευση.



Η εκπαίδευση ολοκληρώνεται πετυχαίνοντας accuracy 0.95 για το training σετ και 0.91 για το validation σετ. Βλέπουμε μία διαφορά ανάμεσα στην ακρίβεια του μοντέλου στο training και validation set. Η διαφορά αυτή μικρή γεγονός που δείχνει ότι το μοντέλο έχει μικρό βαθμό **Overfitting στα δεδομένα εκπαίδευσης**.

Στη συνέχεια ελέγχουμε πως το μοντέλο συμπεριφέρεται στα testing δεδομένα και με τη βοήθεια της συνάρτησης *classification_report()* της Sklearn βλέπουμε τις τιμές των παρακάτω μετρικών για το test set.

Accuracy: 95%

Overall Metrics	**Precision**	**Recall**	**f1-score**
Macro Average	0.96	0.93	0.94
Weighted Average	0.95	0.95	0.95

Metrics Per Class	**Precision**	**Recall**	**f1-score**
Not-Spam	0.93	0.99	0.96
Spam	0.98	0.87	0.92

Στη συνέχεια θα προσπαθήσουμε να βελτιώσουμε τα αποτελέσματα του προηγούμενου δικτύου. Για το λόγο αυτό θα αυξήσουμε τις εποχές εκπαίδευσης από 20 σε 50 καθώς και θα προσθέσουμε περισσότερους Νευρώνες στο κρυφό επίπεδο, ένα επιπλέον κρυφό επίπεδο με συνάρτηση ενεργοποίησης Rectified Linear Unit(ReLU) όπως φαίνεται παρακάτω:

```
[31] # Define the keras model
     model = Sequential()
     model.add(Dense(20, input_dim=vector_size, activation='relu'))
     model.add(Dense(10, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))
```

Πλέον το μοντέλο μας έχει την εξής δομή:

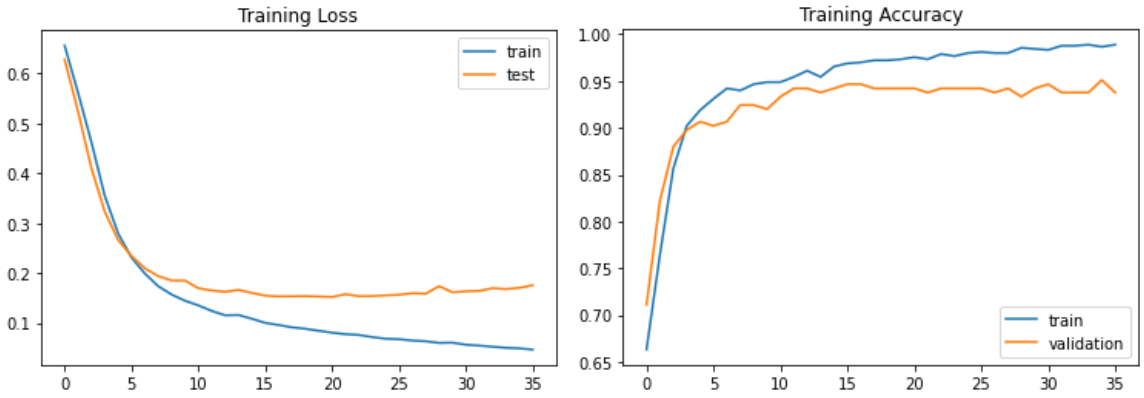
- 1 πρώτο κρυφό επίπεδο 30 Νευρώνων με συνάρτηση ενεργοποίησης Rectified Linear Unit(ReLU).
- 1 δεύτερο κρυφό επίπεδο 15 Νευρώνων με συνάρτηση ενεργοποίησης Rectified Linear Unit(ReLU).
- 1 Νευρώνα στο επίπεδο εξόδου με σιγμοειδή συνάρτηση ενεργοποίησης.
  Επειδή προσθέσαμε ένα ακόμη επίπεδο στο Νευρωνικό μας Δίκτυο αυξάνοντας έτσι και το συνολικό αριθμό Νευρώνων καθώς και τις πιθανότητες το μοντέλο να κάνει overfit στα training data. Για το λόγο αυτό θα εφαρμόσουμε την τεχνική του Early Stopping με τη βοήθεια της συνάρτησης *EarlyStoping()* του **callbacks** module της Keras

```
[81] #Early stopping monitor
     monitor = EarlyStopping(
                              monitor = 'val_loss',
                              min_delta = 1e-3,
                              patience=15,
                              restore_best_weights=True
                              )
```

Δίνουμε τις κατάλληλες παραμέτρους στη συνάρτηση ώστε σε κάθε επανάληψη να ελέγχεται η ποσότητα του σφάλματος στο validation set, ορίζουμε σαν ελάχιστη αλλαγή που μπορεί να θεωρηθεί βελτίωση(min delta) την τιμή 1e-3, στον αριθμό των εποχών που χωρίς βελτίωση η εκπαίδευση τελειώνει(patience) την τιμή 15. Επιπλέον κρατάμε το μοντέλο που είχε την καλύτερη ποσότητα του validation loss.

Εκπαιδεύουμε το μοντέλο κάνοντας πλέον και batch processing με μέγεθος batch 32 ώστε να επιταχύνουμε την εκπαίδευση.

Στις παρακάτω γραφικές φαίνεται η ακρίβεια και το σφάλμα του μοντέλου κατά την εκπαίδευση.



Παρατηρούμε ότι λόγω της χρήσης του Early Stopping η εκπαίδευση ολοκληρώθηκε σε 36 αντί για 50 γύρους.

Πετύχαμε μικρή βελτίωση στο accuracy του μοντέλου με 0.97 στο training σετ και 0.94 στο validation σετ.

Στη συνέχεια ελέγχουμε πως το νέο μοντέλο συμπεριφέρεται στα testing δεδομένα και με τη βοήθεια της συνάρτησης *classification_report()* της Sklearn βλέπουμε τις τιμές των παρακάτω μετρικών για το test set.

Accuracy: 96%

Overall Metrics	**Precision**	**Recall**	**f1-score**
Macro Average	0.97	0.95	0.96
Weighted Average	0.96	0.96	0.96

Metrics Per Class	**Precision**	**Recall**	**f1-score**
Not-Spam	0.95	0.99	0.97
Spam	0.98	0.91	0.95

Παρατηρούμε ότι βελτιώσαμε το προηγούμενο μοντέλο σε μικρό βαθμό, της τάξης του **1-2%** για τις μετρικές που **accuracy, precision, recall, f1-score**, ενώ για κάποιες είχαμε και **βελτίωση της τάξης του 3-4%.**

ΤΕΛΟΣ