EXPRERIMENT 5(a)

IMPLEMENTATION OF LOGISTIC REGRESSION ALGORITHM

NAME : SREENIDHI GANACHARI

REGISTRATION NUMBER : 19BCE7230

SLOT : L-23&24

CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
stroke = pd.read_csv('healthcare-dataset-stroke-data.csv')
stroke.head()
stroke[('stroke')].value_counts()
shuffled_data = stroke.sample(frac=1,random_state=4)
stroke_df = stroke.loc[stroke['stroke'] == 1]
non_stroke_df = stroke.loc[stroke['stroke'] == 0].sample(n= 3500,random
_state= 101)
normalized_stroke = pd.concat([stroke_df, non_stroke_df])
sns.countplot('stroke', data= normalized_stroke, palette= "colorblind")
plt.title('Stroke Analysis')
plt.show()
sns.countplot(x='stroke', hue = 'gender', data = normalized_stroke, pal
ette = "Set1")
plt.title('Gender Split')
plt.show()
plt.figure(figsize=(8,7))
sns.boxplot(x = 'stroke', y = 'bmi', hue = 'gender', data= normalized_s
troke, palette= "winter")
plt.title('Subject BMIs')
plt.show()
sns.heatmap(normalized_stroke.isnull(), yticklabels=False, cbar=False,
cmap='viridis')
def input_bmi(cols):
    bmi = cols[0]
    stroke = cols [1]


    if pd.isnull(bmi):
        return 28.6
```

```
    else:
        return bmi
normalized_stroke['bmi'] = stroke[['bmi', 'stroke']].apply(input_bmi, a
xis=1)
sns.heatmap(normalized_stroke.isnull(), yticklabels=False, cbar=False,
cmap='viridis')
sns.countplot(x='stroke', hue = 'Residence_type', data =normalized_stro
ke, palette = 'GnBu')
plt.title('Residence Type')
plt.show()
sns.countplot(x='ever_married', hue = 'stroke', data = normalized_strok
e)
plt.title('Marital Status')
plt.show()
sns.countplot(x='hypertension', hue = 'stroke', data = normalized_strok
e)
plt.title('Hypertension Check')
plt.show()
sns.countplot(x='heart_disease', hue = 'stroke', data = normalized_stro
ke)
plt.title('Heart Condition')
plt.show()
sns.countplot(x='work_type', hue = 'stroke', data = normalized_stroke)
plt.title('Occupation')
plt.show()
sns.barplot(x='stroke', y = 'avg_glucose_level', data = normalized_stro
ke)
plt.title('Blood Glucose Level')
plt.show()
residence = pd.get_dummies(normalized_stroke['Residence_type'])
residence.head()
residence = pd.get_dummies(normalized_stroke["Residence_type"], drop_fi
rst= True)
normalized_stroke.drop(["Residence_type"], axis = 1, inplace = True)
normalized_stroke = pd.concat([normalized_stroke, residence], axis = 1)
normalized_stroke.head()
normalized_stroke.rename(columns={'Urban':'Residence_type'},
                 inplace=True)
sex = pd.get_dummies(normalized_stroke['gender'])
sex = pd.get_dummies(normalized_stroke["gender"], drop_first= True)
normalized_stroke.drop(["gender"], axis = 1, inplace = True)
normalized_stroke = pd.concat([normalized_stroke, sex], axis = 1)
marital_status = pd.get_dummies(normalized_stroke['ever_married'])
marital_status = pd.get_dummies(normalized_stroke["ever_married"], drop
_first= True)
normalized_stroke.drop(["ever_married", "smoking_status"], axis = 1, in
place = True)
```

```
normalized_stroke = pd.concat([normalized_stroke, marital_status], axis
 = 1)
normalized_stroke.rename(columns={'Yes':'marital_status'},
                 inplace=True)


occupation = pd.get_dummies(normalized_stroke['work_type'])
normalized_stroke.drop(["work_type"], axis = 1, inplace = True)
normalized_stroke = pd.concat([normalized_stroke, occupation], axis = 1
)
normalized_stroke.drop(["avg_glucose_level"], axis = 1, inplace = True)
normalized_stroke.head()
normalized_stroke.drop(["id"], axis = 1, inplace = True)
from sklearn.model_selection import train_test_split
X = normalized_stroke.drop('stroke', axis = 1)
y = normalized_stroke['stroke']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.3, random_state=101)
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
predictions = logmodel.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
logmodel.score(X_test, y_test)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,predictions))
sns.heatmap(confusion_matrix(y_test,predictions), annot= True, cmap = '
viridis', fmt="2")
plt.title('Confusion Matrix')
plt.show()
```

OUTPUT –

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
```

```
[ ] stroke = pd.read_csv('healthcare-dataset-stroke-data.csv')
    stroke.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

```
[ ] stroke[('stroke')].value_counts()
```

```
0    4861
1     249
Name: stroke, dtype: int64
```

```
shuffled_data = stroke.sample(frac=1,random_state=4)
stroke_df = stroke.loc[stroke['stroke'] == 1]
non_stroke_df = stroke.loc[stroke['stroke'] == 0].sample(n= 3500,random_state= 101)
```

Type here to search

ENG 19:24 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

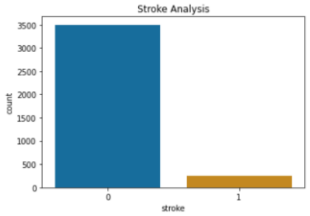
+ Code + Text

Connect Editing

```
[ ] normalized_stroke = pd.concat([stroke_df, non_stroke_df])
```

```
sns.countplot('stroke', data= normalized_stroke, palette= "colorblind")
plt.title('Stroke Analysis')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument
  FutureWarning
```




```
sns.countplot(x='stroke', hue = 'gender', data = normalized_stroke, palette = "Set1")
plt.title('Gender Split')
plt.show()
```

Type here to search

ENG 19:25 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

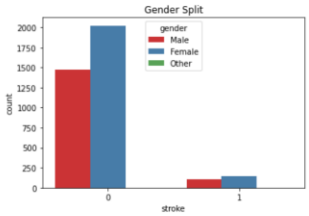
Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

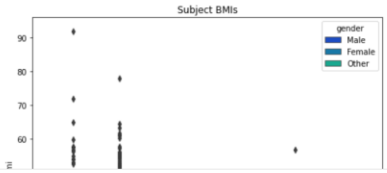
Comment Share

+ Code + Text

Connect Editing




```
plt.figure(figsize=(8,7))
sns.boxplot(x = 'stroke', y = 'bmi', hue = 'gender', data= normalized_stroke, palette= "winter")
plt.title('Subject BMIs')
plt.show()
```



Type here to search

ENG 19:25 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

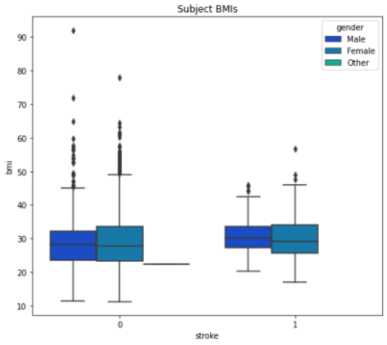
Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing




```
sns.heatmap(normalized_stroke.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb0ae91550>
```

Type here to search

ENG 19:25 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

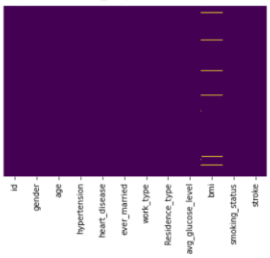
Comment Share

+ Code + Text

Connect Editing

```
sns.heatmap(normalized_stroke.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb0ae91550>
```




```
[ ] def input_bmi(cols):
        bmi = cols[0]
        stroke = cols [1]


        if pd.isnull(bmi):
            return 28.6
        else:
            return bmi
```

Type here to search

ENG 19:25 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

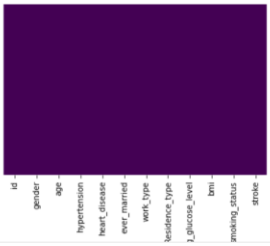
Connect Editing

```
[ ] def input_bmi(cols):
        bmi = cols[0]
        stroke = cols [1]


        if pd.isnull(bmi):
            return 28.6
        else:
            return bmi
```

```
[ ] normalized_stroke['bmi'] = stroke[['bmi', 'stroke']].apply(input_bmi, axis=1)
```

```
sns.heatmap(normalized_stroke.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffb084fd590>
```



Type here to search

ENG 19:25 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

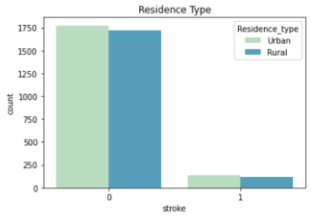
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

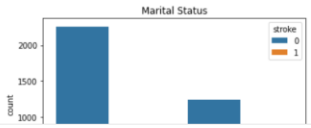
+ Code + Text

Connect Editing

```
[ ] sns.countplot(x='stroke', hue = 'Residence_type', data =normalized_stroke, palette = 'GnBu')
    plt.title('Residence Type')
    plt.show()
```




```
sns.countplot(x='ever_married', hue = 'stroke', data = normalized_stroke)
plt.title('Marital Status')
plt.show()
```



Type here to search

ENG 19:26 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

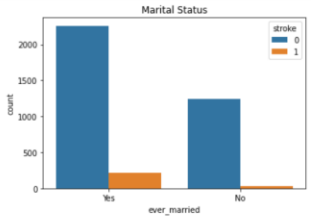
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

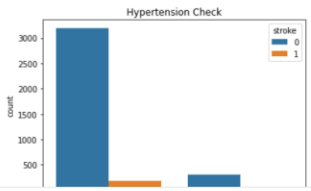
+ Code + Text

Connect Editing

[ ]




```
sns.countplot(x='hypertension', hue = 'stroke', data = normalized_stroke)
plt.title('Hypertension Check')
plt.show()
```



Type here to search

ENG 19:26 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

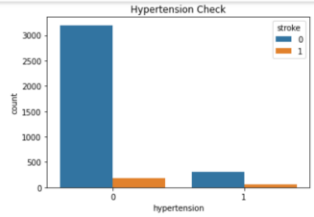
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

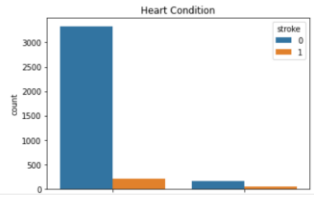
+ Code + Text

Connect Editing

[ ]




```
sns.countplot(x='heart_disease', hue = 'stroke', data = normalized_stroke)
plt.title('Heart Condition')
plt.show()
```



Type here to search

ENG 19:26 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=AK7Hn1G_zUhl

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

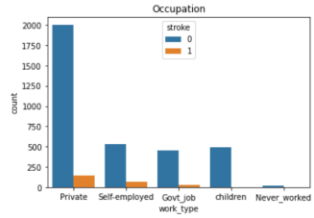
Comment Share

+ Code + Text

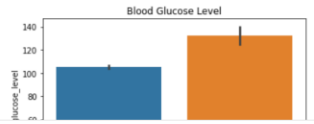
Connect Editing

[ ]

```
sns.countplot(x='work_type', hue = 'stroke', data = normalized_stroke)
plt.title('Occupation')
plt.show()
```

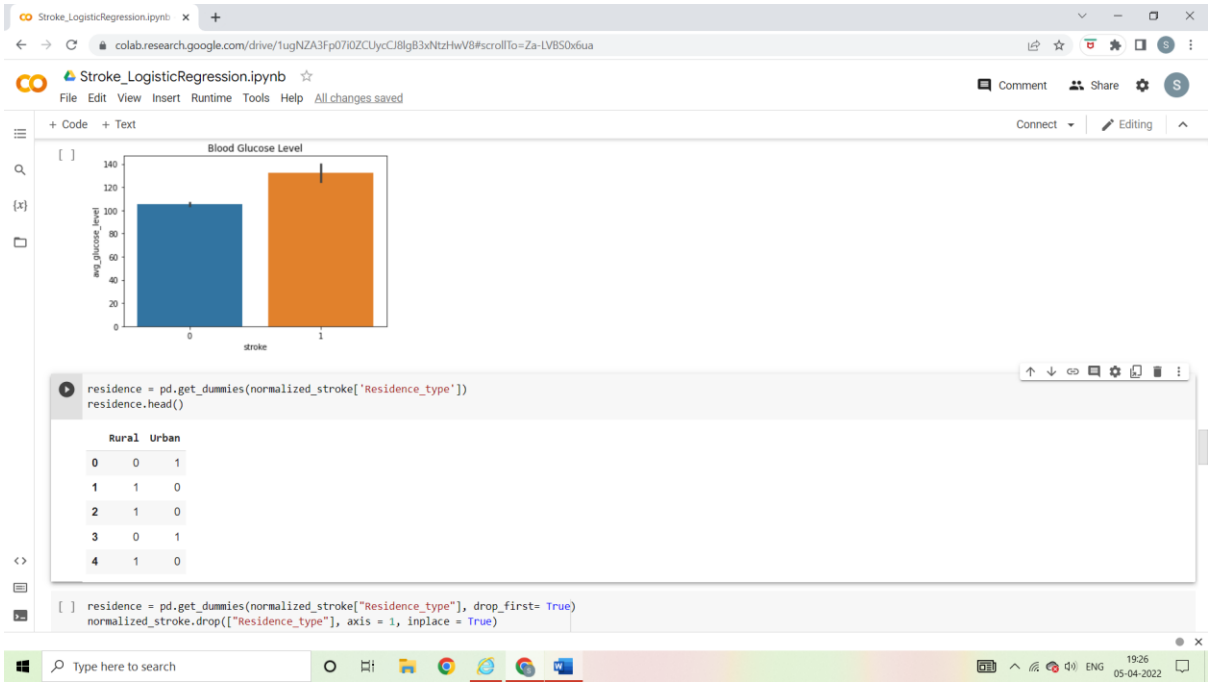


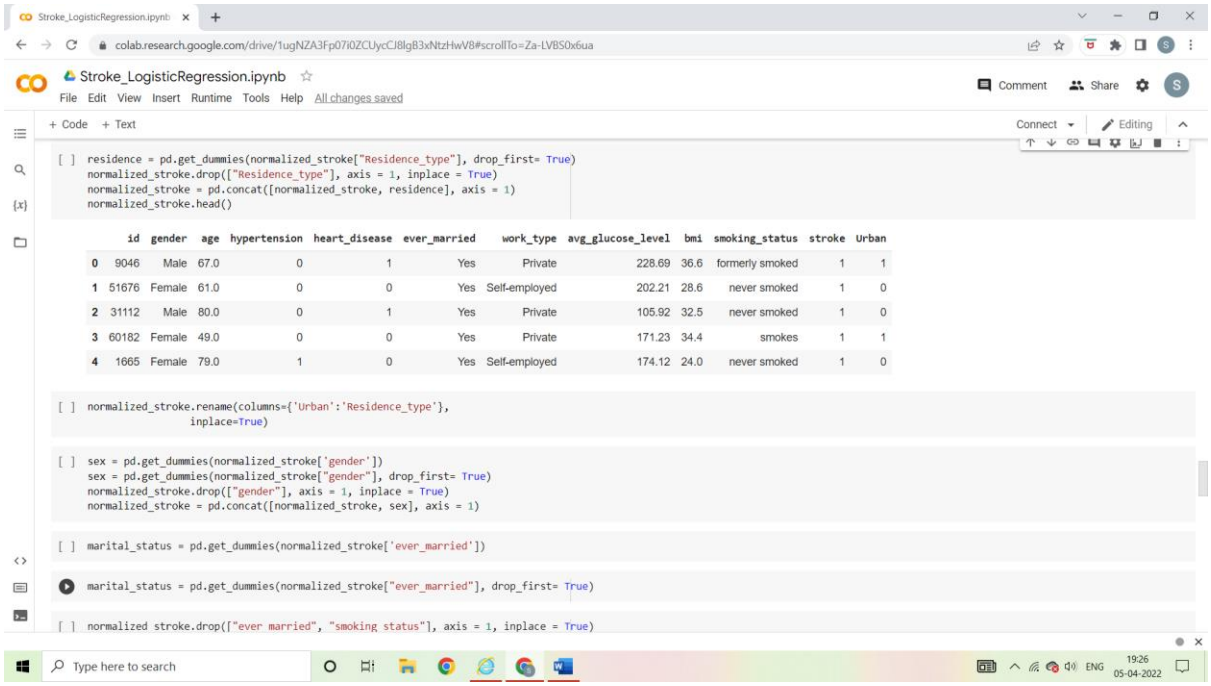

```
sns.barplot(x='stroke', y = 'avg_glucose_level', data = normalized_stroke)
plt.title('Blood Glucose Level')
plt.show()
```

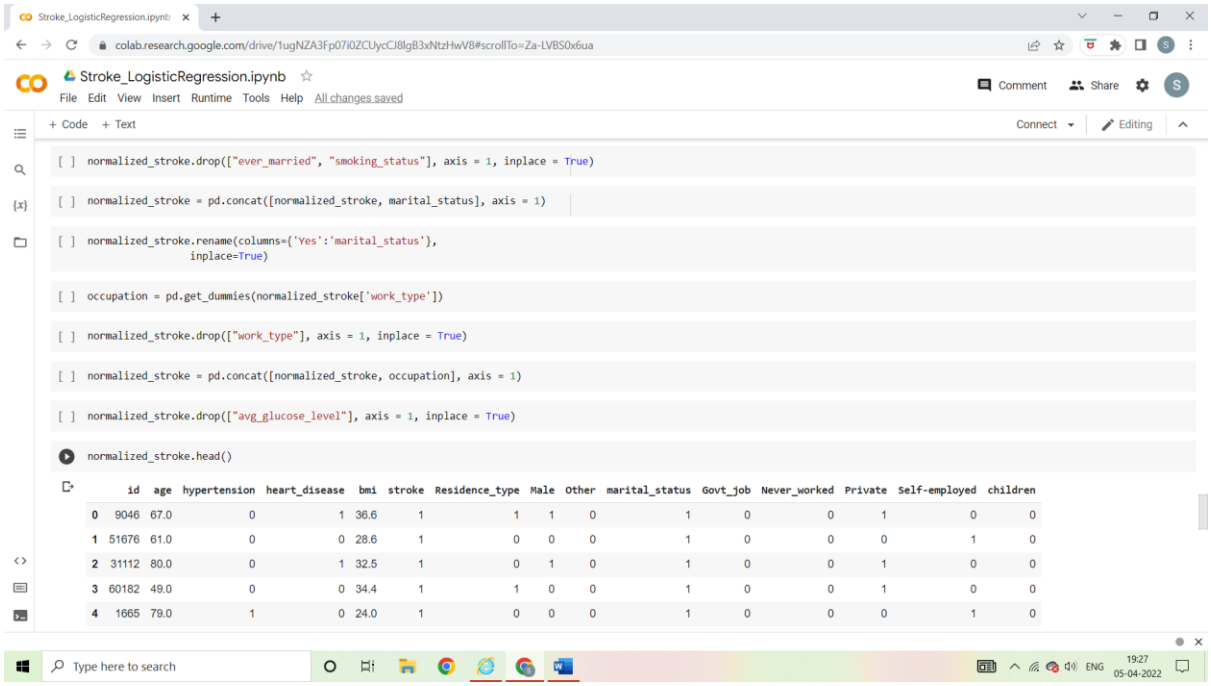


Type here to search

ENG 19:26 05-04-2022


Stroke_LogisticRegression.ipynb
colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua
Stroke_LogisticRegression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
Blood Glucose Level
avg_glucose_level
stroke
residence = pd.get_dummies(normalized_stroke['Residence_type'])
residence.head()
Rural Urban
0 0 1
1 1 0
2 1 0
3 0 1
4 1 0
residence = pd.get_dummies(normalized_stroke["Residence_type"], drop_first= True)
normalized_stroke.drop(["Residence_type"], axis = 1, inplace = True)
Type here to search
ENG
19:26
05-04-2022


Stroke_LogisticRegression.ipynb
colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua
Stroke_LogisticRegression.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
residence = pd.get_dummies(normalized_stroke["Residence_type"], drop_first= True)
normalized_stroke.drop(["Residence_type"], axis = 1, inplace = True)
normalized_stroke = pd.concat([normalized_stroke, residence], axis = 1)
normalized_stroke.head()
id gender age hypertension heart_disease ever_married work_type avg_glucose_level bmi smoking_status stroke Urban
0 9046 Male 67.0 0 1 Yes Private 228.69 36.6 formerly smoked 1 1
1 51676 Female 61.0 0 0 Yes Self-employed 202.21 28.6 never smoked 1 0
2 31112 Male 80.0 0 1 Yes Private 105.92 32.5 never smoked 1 0
3 60182 Female 49.0 0 0 Yes Private 171.23 34.4 smokes 1 1
4 1665 Female 79.0 1 0 Yes Self-employed 174.12 24.0 never smoked 1 0
normalized_stroke.rename(columns={'Urban':'Residence_type'},
inplace=True)
sex = pd.get_dummies(normalized_stroke['gender'])
sex = pd.get_dummies(normalized_stroke["gender"], drop_first= True)
normalized_stroke.drop(["gender"], axis = 1, inplace = True)
normalized_stroke = pd.concat([normalized_stroke, sex], axis = 1)
marital_status = pd.get_dummies(normalized_stroke['ever_married'])
marital_status = pd.get_dummies(normalized_stroke["ever_married"], drop_first= True)
normalized stroke.drop(["ever married", "smoking status"], axis = 1, inplace = True)
Type here to search
ENG
19:26
05-04-2022



Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

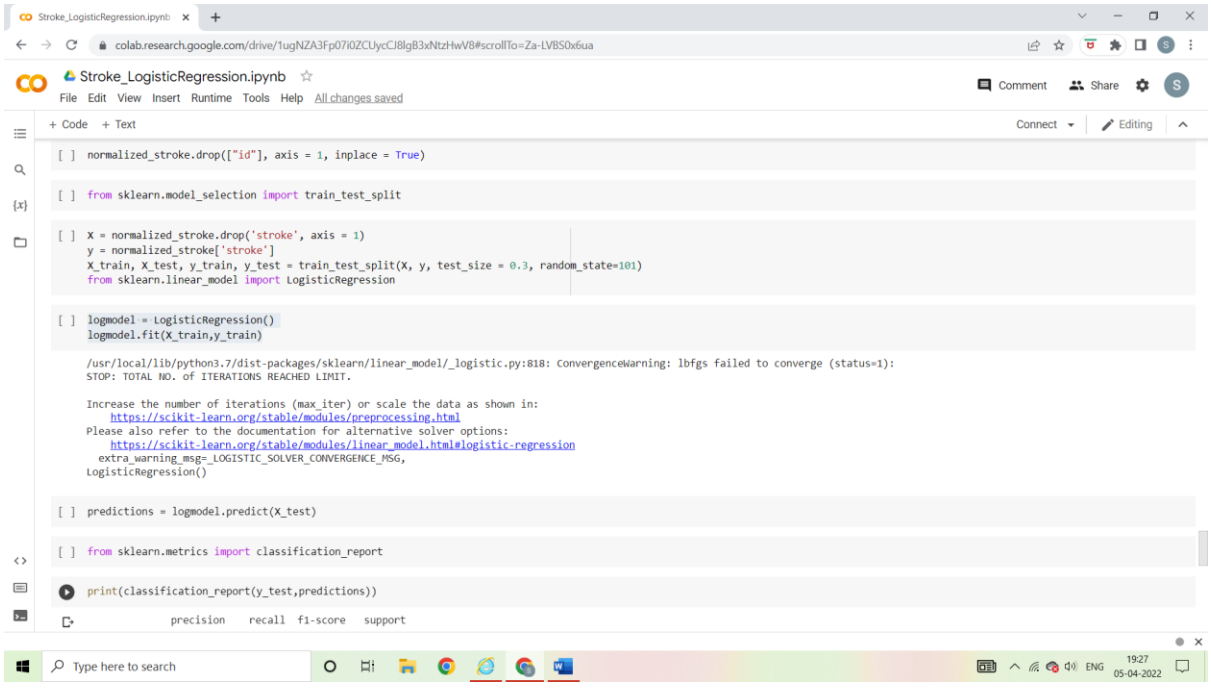
Connect Editing

```
[ ] normalized_stroke.drop(["ever_married", "smoking_status"], axis = 1, inplace = True)

[ ] normalized_stroke = pd.concat([normalized_stroke, marital_status], axis = 1)

[ ] normalized_stroke.rename(columns={'Yes':'marital_status'},
                             inplace=True)

[ ] occupation = pd.get_dummies(normalized_stroke['work_type'])

[ ] normalized_stroke.drop(["work_type"], axis = 1, inplace = True)

[ ] normalized_stroke = pd.concat([normalized_stroke, occupation], axis = 1)

[ ] normalized_stroke.drop(["avg_glucose_level"], axis = 1, inplace = True)

normalized_stroke.head()
```

	id	age	hypertension	heart_disease	bmi	stroke	Residence_type	Male	Other	marital_status	Govt_job	Never_worked	Private	Self-employed	children
0	9046	67.0	0	1	36.6	1	1	1	0	1	0	0	1	0	0
1	51676	61.0	0	0	28.6	1	0	0	0	1	0	0	0	1	0
2	31112	80.0	0	1	32.5	1	0	1	0	1	0	0	1	0	0
3	60182	49.0	0	0	34.4	1	1	0	0	1	0	0	1	0	0
4	1665	79.0	1	0	24.0	1	0	0	0	1	0	0	0	1	0

Type here to search

ENG 19:27 05-04-2022



Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
[ ] normalized_stroke.drop(["id"], axis = 1, inplace = True)

[ ] from sklearn.model_selection import train_test_split

[ ] X = normalized_stroke.drop('stroke', axis = 1)
    y = normalized_stroke['stroke']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=101)
    from sklearn.linear_model import LogisticRegression

[ ] logmodel = LogisticRegression()
    logmodel.fit(X_train,y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()

[ ] predictions = logmodel.predict(X_test)

[ ] from sklearn.metrics import classification_report

print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support
```

Type here to search

ENG 19:27 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

```
[ ] from sklearn.metrics import classification_report
```

```
[ ] print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96      1046
           1       0.00      0.00      0.00        79

    accuracy                           0.93      1125
   macro avg       0.46      0.50      0.48      1125
weighted avg       0.86      0.93      0.90      1125
```

```
[ ] logmodel.score(X_test, y_test)
```

```
0.9288888888888889
```

```
[ ] from sklearn.metrics import confusion_matrix
```

```
[ ] print(confusion_matrix(y_test,predictions))
```

```
[[1045    1]
 [  79    0]]
```

```
[ ] sns.heatmap(confusion_matrix(y_test,predictions), annot= True, cmap = 'viridis', fmt="2")
    plt.title('Confusion Matrix')
    plt.show()
```

Type here to search

ENG 19:27 05-04-2022

Stroke_LogisticRegression.ipynb

colab.research.google.com/drive/1ugNZA3Fp07i0ZCUycCJ8lgB3xNtzHwV8#scrollTo=Za-LVBS0x6ua

Stroke_LogisticRegression.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Editing

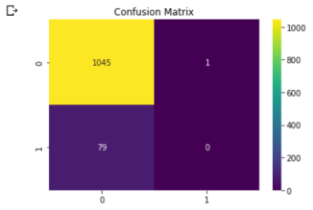
0.9288888888888889

[ ] from sklearn.metrics import confusion_matrix

[ ] print(confusion_matrix(y_test,predictions))

[[1045 1]
[79 0]]

sns.heatmap(confusion_matrix(y_test,predictions), annot= True, cmap = 'viridis', fmt="2")
plt.title('Confusion Matrix')
plt.show()



Type here to search

ENG 19:27 05-04-2022