Task 4- Model Buildings.ipynb

colab.research.google.com/drive/1IMogoQr7cOS7sHAI-K9Sqr4yir5BoeeN#scrollTo=xA4uOA2kha3p&uniqifier=9

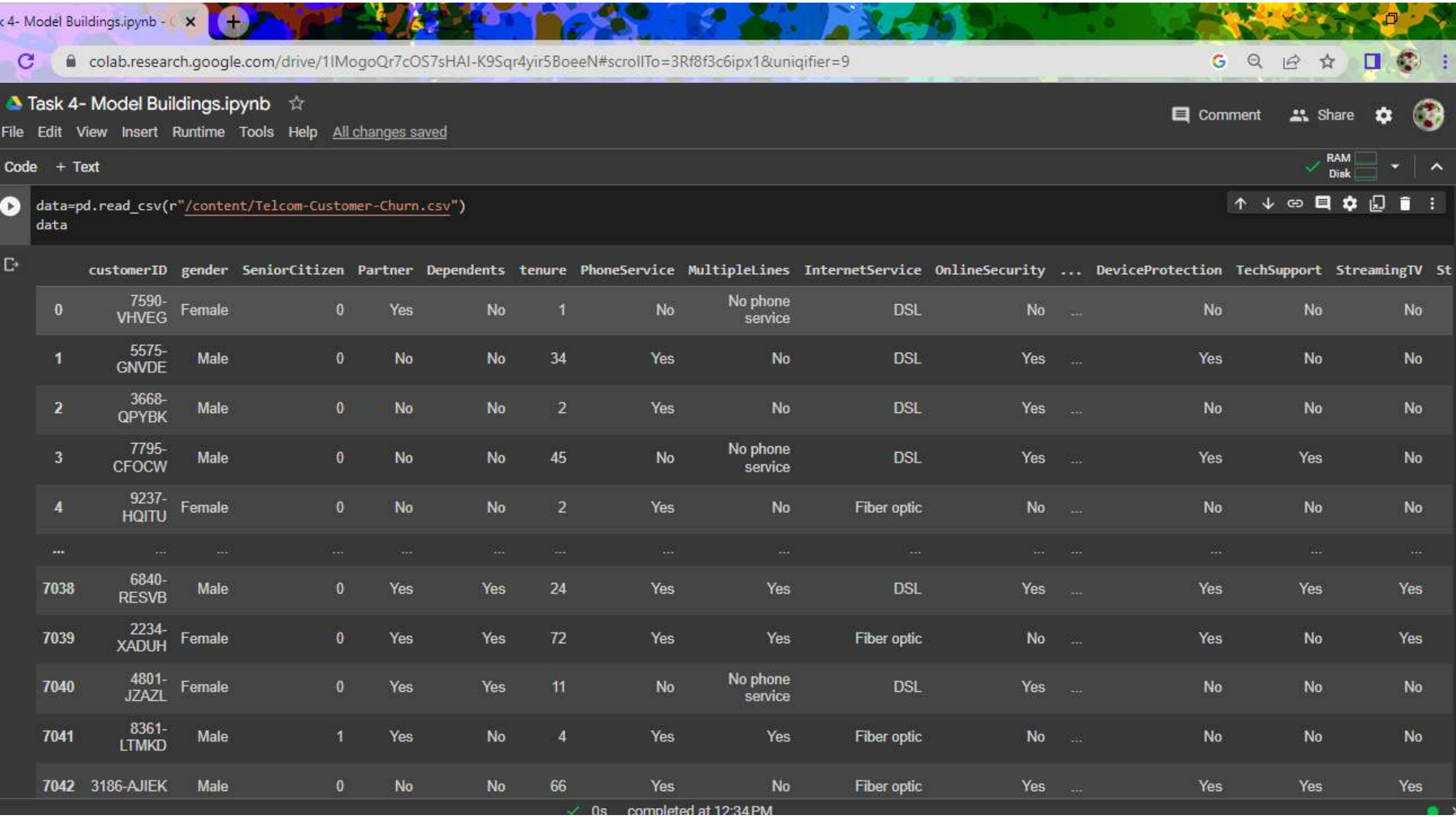
Task 4- Model Buildings.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

0s

```
#import necessary libraries
import pandas as pd
import numpy as np
import pickle
import matplotlib. pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report, confusion_matrix,f1_score
```



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	St
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	No	No	
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No	No	No	
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	Yes	Yes	Yes	

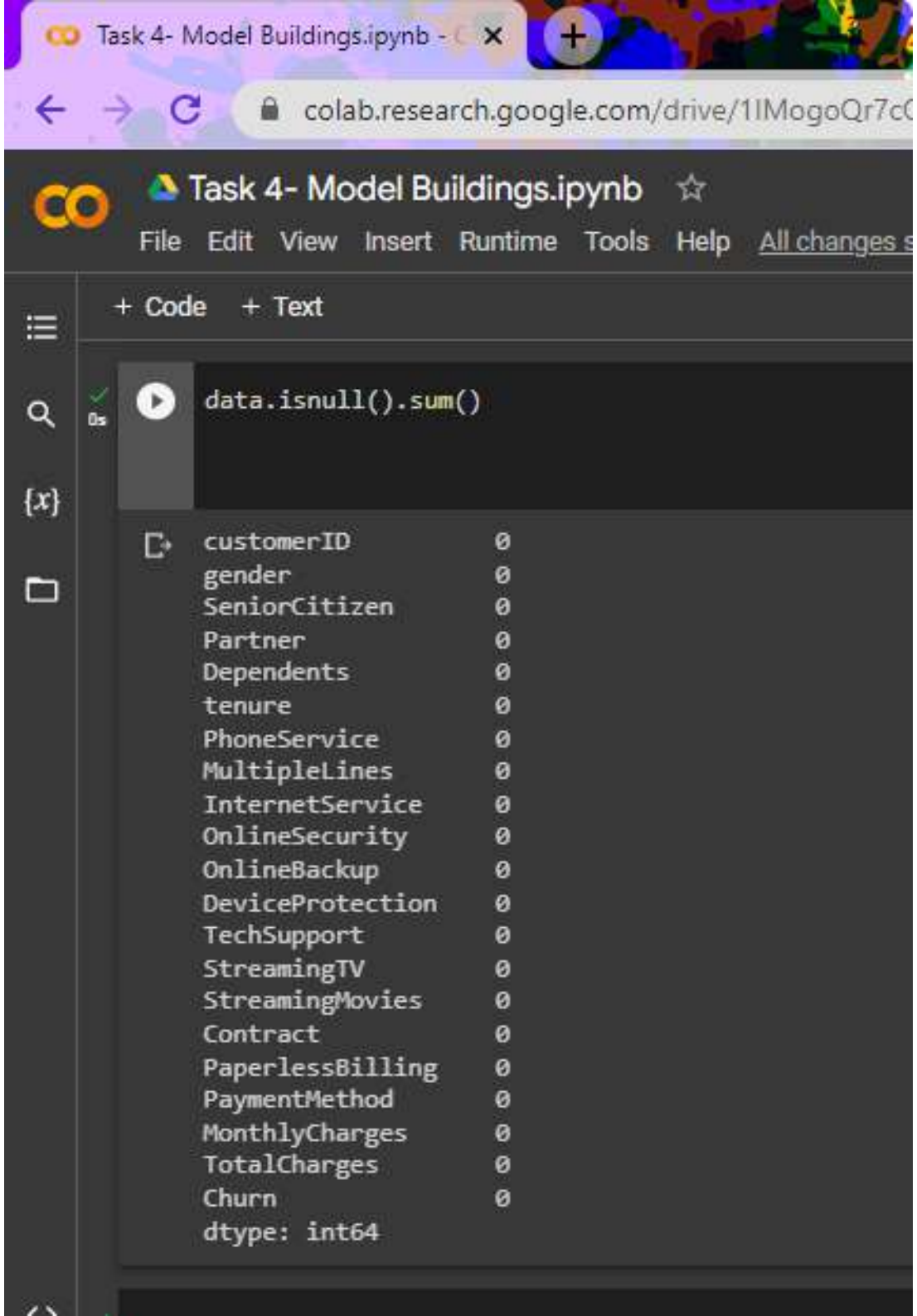
Task 4- Model Buildings.ipynb

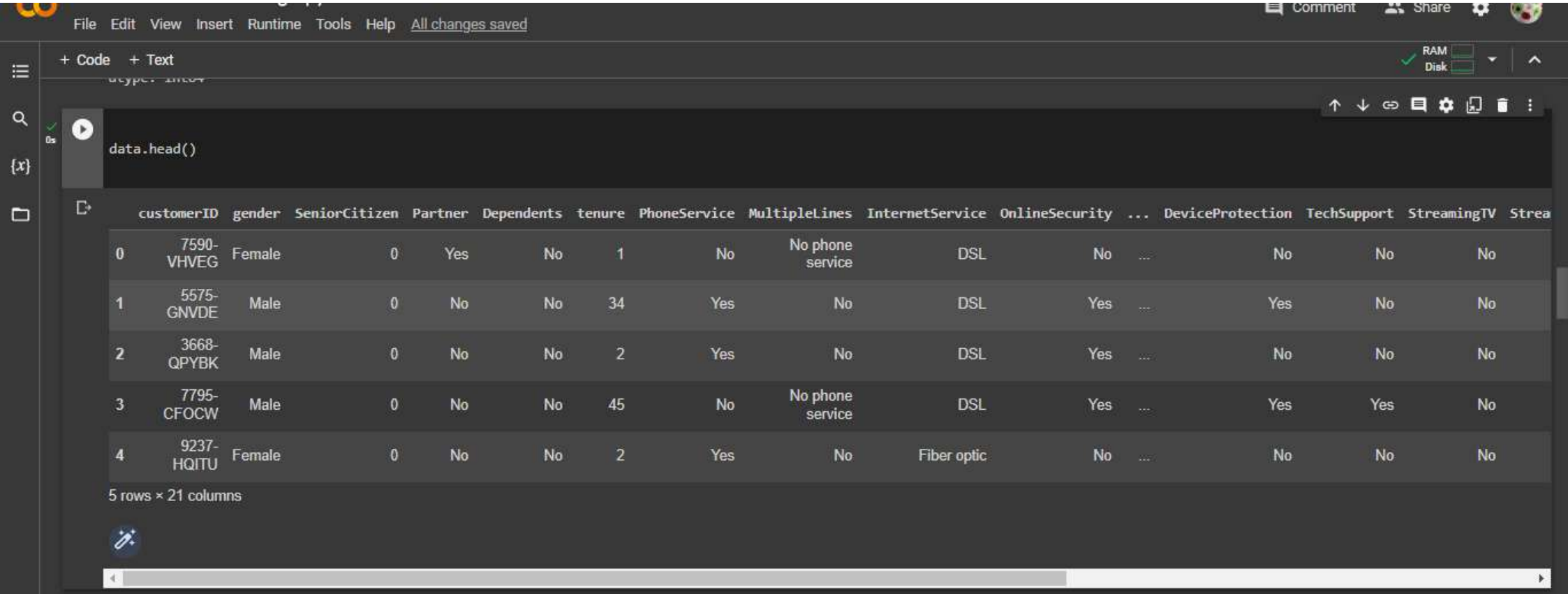
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[18] data.info()
```

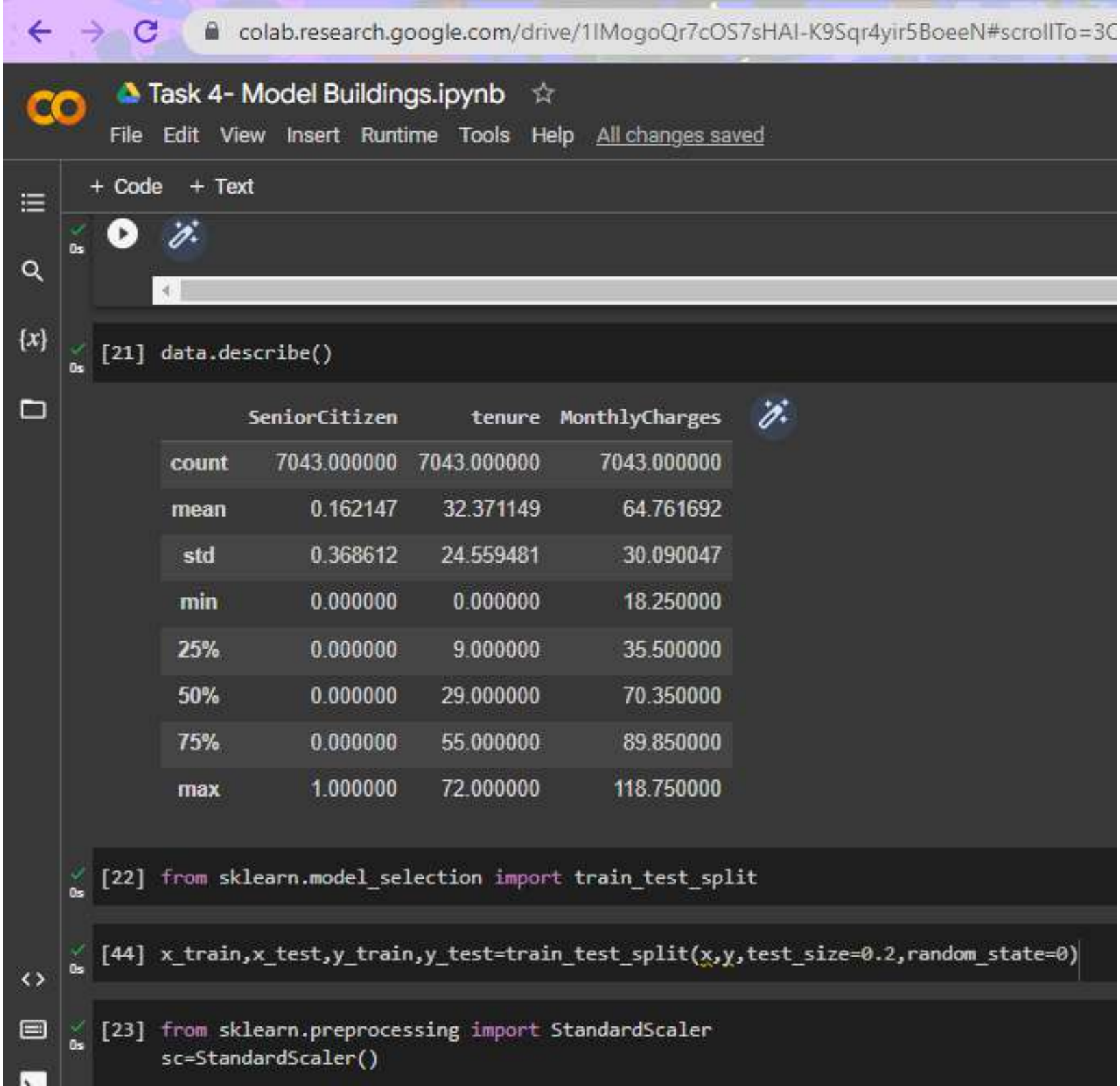
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```


Task 4- Model Buildings.ipynb
colab.research.google.com/drive/1IMogoQr7c
Task 4- Model Buildings.ipynb
File Edit View Insert Runtime Tools Help All changes s
+ Code + Text
0s
data.isnull().sum()
customerID 0
gender 0
SeniorCitizen 0
Partner 0
Dependents 0
tenure 0
PhoneService 0
MultipleLines 0
InternetService 0
OnlineSecurity 0
OnlineBackup 0
DeviceProtection 0
TechSupport 0
StreamingTV 0
StreamingMovies 0
Contract 0
PaperlessBilling 0
PaymentMethod 0
MonthlyCharges 0
TotalCharges 0
Churn 0
dtype: int64

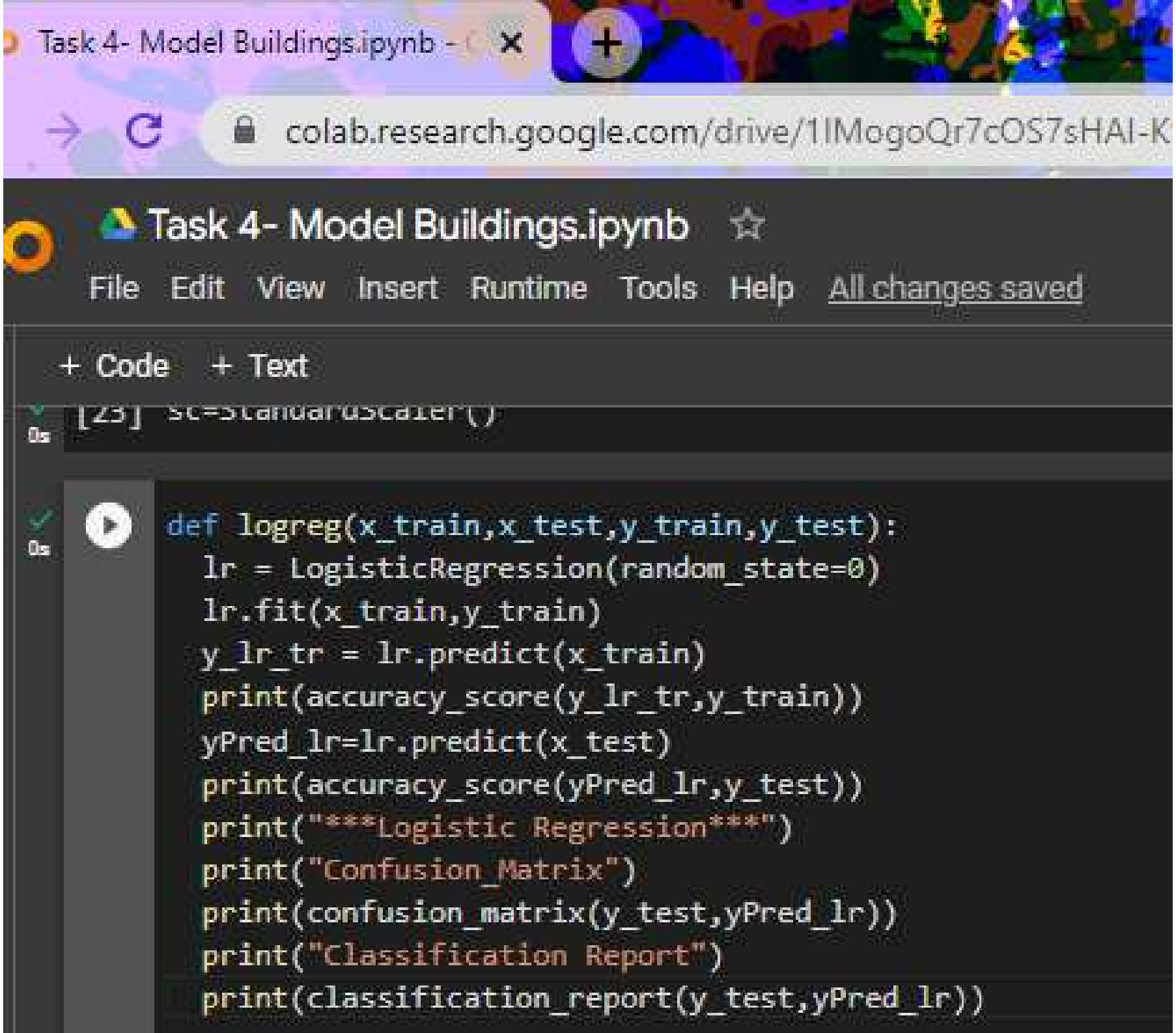


	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	Strea
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	

5 rows × 21 columns



	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000


Task 4- Model Buildings.ipynb
colab.research.google.com/drive/1IMogoQr7cOS7sHAI-K
Task 4- Model Buildings.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s
[23] sc=StandardScaler()
0s
def logreg(x_train,x_test,y_train,y_test):
lr = LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
y_lr_tr = lr.predict(x_train)
print(accuracy_score(y_lr_tr,y_train))
yPred_lr=lr.predict(x_test)
print(accuracy_score(yPred_lr,y_test))
print("***Logistic Regression***")
print("Confusion Matrix")
print(confusion_matrix(y_test,yPred_lr))
print("Classification Report")
print(classification_report(y_test,yPred_lr))

```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```

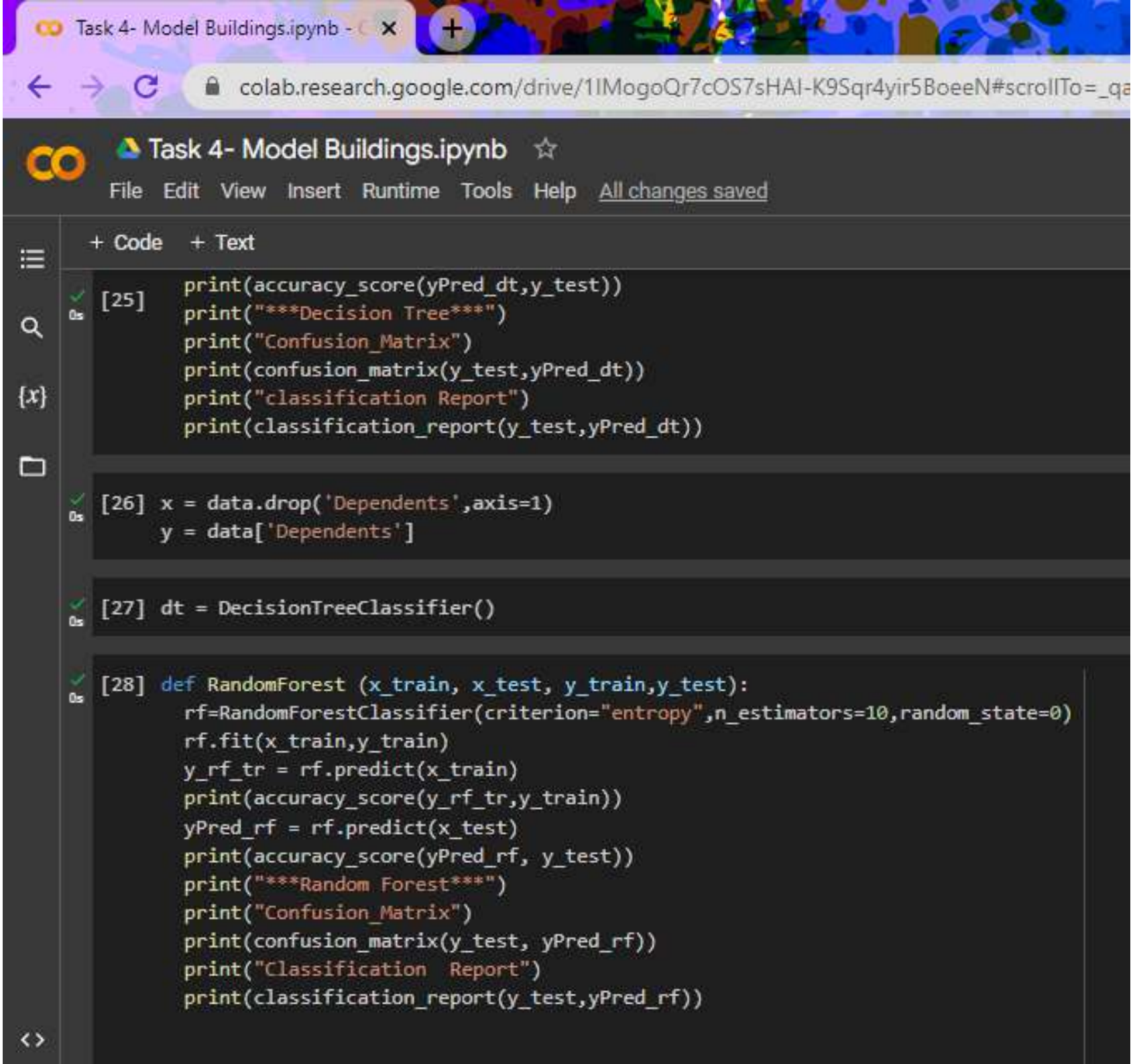
```python
def decisionTree(x_train,x_test,y_train,y_test):
    dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    y_dt_tr = dtc.predict(x_train)
    print(accuracy_score(y_dt_tr,y_train))
    yPred_dt = dtc.predict(x_test)
    print(accuracy_score(yPred_dt,y_test))
    print("***Decision Tree***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_dt))
    print("Classification Report")
    print(classification_report(y_test,yPred_dt))
```

```python
#printing the train accuracy and test accuracy respectively
decisionTree(x_train,x_test,y_train,y_test)
```

```
0.9981879681082387
0.6067632850241546
***Decision Tree***
Confusion_Matrix
[[ 242  791]
 [  23 1014]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.23      0.37      1033
           1       0.56      0.98      0.71      1037

    accuracy                           0.61      2070
   macro avg       0.74      0.61      0.54      2070
```


Task 4- Model Buildings.ipynb
colab.research.google.com/drive/1IMogoQr7cOS7sHAI-K9Sqr4yir5BoeeN#scrollTo=_qa
Task 4- Model Buildings.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[25] print(accuracy_score(yPred_dt,y_test))
print("***Decision Tree***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,yPred_dt))
print("classification Report")
print(classification_report(y_test,yPred_dt))
[26] x = data.drop('Dependents',axis=1)
y = data['Dependents']
[27] dt = DecisionTreeClassifier()
[28] def RandomForest (x_train, x_test, y_train,y_test):
rf=RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
rf.fit(x_train,y_train)
y_rf_tr = rf.predict(x_train)
print(accuracy_score(y_rf_tr,y_train))
yPred_rf = rf.predict(x_test)
print(accuracy_score(yPred_rf, y_test))
print("***Random Forest***")
print("Confusion_Matrix")
print(confusion_matrix(y_test, yPred_rf))
print("Classification Report")
print(classification_report(y_test,yPred_rf))

```
RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9886446001449626
0.7536231884057971
***Random Forest***
Confusion Matrix
[[563 470]
 [ 40 997]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.55      0.69      1033
           1       0.68      0.96      0.80      1037

    accuracy                           0.75      2070
   macro avg       0.81      0.75      0.74      2070
weighted avg       0.81      0.75      0.74      2070
```

```
] def KNN(x_train, x_test,y_train,y_test) :
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    y_knn_tr = knn.predict(x_train)
    print(accuracy_score (y_knn_tr,y_train) )
    yPred_knn = knn.predict(x_test)
    print(accuracy_score (yPred_knn,y_test))
    print("***KNN***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test, yPred_knn))
    print("Classification Report")
    print(classification_report(y_test,yPred_knn))
```

```
KNN(x_train,x_test,y_train,y_test)
```

```
0.8570910848030925
0.7913043478260869
***KNN***
Confusion Matrix
[[730 303]
 [129 908]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.71      0.77      1033
           1       0.75      0.88      0.81      1037

    accuracy                           0.79      2070
   macro avg       0.80      0.79      0.79      2070
weighted avg       0.80      0.79      0.79      2070
```

```
0] def svm(x_tarin,x_test,y_train,y_test):
    svm = SVC(kernel ="linear")
    svm.fit(x_train,y_train)
    y_svm_tr = svm. predict (x_train)
    print(accuracy_score(y_svm_tr,y_train))
    yPred_svm = svm. predict (x_test)
    print(accuracy_score(yPred_svm,y_test))
    print("***Support Vector Machine***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_svm))
    print("Classification Report")
    print(classification_report(y_test,yPred_svm))
```

```
svm(x_train,x_test,y_train,y_test)
```

```
0.7628654264315052
0.7555555555555555
***Support Vector Machine***
Confusion_Matrix
[[719 314]
 [192 845]]
Classification Report
              precision    recall  f1-score   support

           0       0.79      0.70      0.74      1033
           1       0.73      0.81      0.77      1037

    accuracy                           0.76      2070
   macro avg       0.76      0.76      0.75      2070
```

```
[32] import keras
     from keras.models import Sequential
     from keras.layers import Dense
```

```
[33] classifier = Sequential()
```

```
[34] #adding the input layer and first hiden layer
     classifier.add(Dense(units=30,activation='relu',input_dim=40))
```

```
[35] #adding the second hiden layer
     classifier.add(Dense(units=30,activation='relu'))
```

```
[37] #adding the output layer
     classifier.add(Dense(units=1,activation='sigmoid'))
```

```
[39] #compiling the ANN
     classifier.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
```

```
[59] from sklearn.preprocessing import LabelEncoder
```

```
[61] le=LabelEncoder()
```

```
model_history = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=200)

Epoch 1/200
555/555 [==============================] - 4s 3ms/step - loss: 0.5017 - accuracy: 0.7494 - val_loss: 0.4688 - val_accuracy: 0.775
Epoch 2/200
555/555 [==============================] - 2s 3ms/step - loss: 0.4535 - accuracy: 0.7815 - val_loss: 0.4627 - val_accuracy: 0.778
Epoch 3/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4424 - accuracy: 0.7865 - val_loss: 0.4691 - val_accuracy: 0.777
Epoch 4/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4325 - accuracy: 0.7950 - val_loss: 0.4541 - val_accuracy: 0.791
Epoch 5/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4239 - accuracy: 0.8002 - val_loss: 0.4536 - val_accuracy: 0.789
Epoch 6/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4146 - accuracy: 0.8078 - val_loss: 0.4564 - val_accuracy: 0.793
Epoch 7/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4058 - accuracy: 0.8100 - val_loss: 0.4551 - val_accuracy: 0.792
Epoch 8/200
```

```
ann_pred = (ann_pred>0.5)
ann_pred
```

```
65/65 [==============================] - 0s 2ms/step
array([[False],
       [False],
       [ True],
       ...,
       [False],
       [False],
       [False]])
```

```
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,ann_pred))
print("Classification Report")
print(classification_report(y_test,ann_pred))
```

```
0.8067632850241546
***ANN Model***
Confusion_Matrix
[[840 193]
 [207 830]]
Classification Report
              precision    recall  f1-score   suppor
```

```
lr = LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
print("Predicting on random input")
lr_pred_own = lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",lr_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
dtc.fit(x_train,y_train)
print("Predicting on random input")
dtc_pred_own = dtc.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",dtc_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
rf.fit(x_train,y_train)
print("Predicting on random input")
rf_pred_own = rf.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",rf_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
svc = SVC(kernel = "linear")
svc.fit(x_train,y_train)
print("Predicting on random input")
svm_pred_own = svc.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",svm_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
print("Predicting on random input")
knn_pred_own = knn.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",knn_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```python
#testing on random input values
print("Predicting on random input")
ann_pred_own = classifier.predict(sc.transform([[0,0,1,1,0,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print(ann_pred_own)
ann_pred_own = (ann_pred_own>0.5)
print("output is: ",ann_pred_own)
```

```
Predicting on random input
1/1 [==============================] - 0s 24ms/step
[[1.]]
output is:  [[ True]]
```