SOURCE CODE

SGD OPTIMIZATION

```
from google.colab import drive

drive.mount('/content/gdrive')

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split


d1=pd.read_csv('/content/gdrive/MyDrive/dataset-org.csv',header=None)
print(d1)


X=pd.DataFrame(d1.iloc[:,:-1].values)
Y=d1.iloc[:,-1].values


X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 0 )


print(X_train)


print(X_test)


print(Y_train)


print(Y_test)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

import tensorflow as tf

import keras  as ke

from keras.models import Sequential
```

```
from keras.layers import Dense

classifier = Sequential()
classifier.add(Dense(units= 16,kernel_initializer='uniform',activation='relu',input_dim=178))


classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 1,kernel_initializer   = 'uniform', activation = 'sigmoid'))


classifier.compile(optimizer = 'sgd', loss = 'binary_crossentropy', metrics = ['accuracy'])

classifier.fit(X_train, Y_train, batch_size = 10, epochs = 100)

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

y_pred

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test,y_pred)
```

ADAGRAD OPTIMIZATION

```
from google.colab import drive

drive.mount('/content/gdrive')

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split


d1=pd.read_csv('/content/gdrive/MyDrive/dataset-org.csv',header=None)
print(d1)


X=pd.DataFrame(d1.iloc[:,:-1].values)
```

```
Y=d1.iloc[:,-1].values

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 0 )

print(X_train)

print(X_test)

print(Y_train)

print(Y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

import tensorflow as tf

import keras  as ke

from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(units=16,kernel_initializer='uniform',activation='relu',input_dim=178))


classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 1,kernel_initializer   = 'uniform', activation = 'sigmoid'))

classifier.compile(optimizer = 'adagrad', loss = 'binary_crossentropy', metrics = ['accuracy'])

classifier.fit(X_train, Y_train, batch_size = 10, epochs = 100)

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

```
y_pred

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test,y_pred)
```

FTRL OPTIMIZATION

```
from google.colab import drive

drive.mount('/content/gdrive')


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split


d1=pd.read_csv('/content/gdrive/MyDrive/dataset-org.csv',header=None)
print(d1)


X=pd.DataFrame(d1.iloc[:,:-1].values)
Y=d1.iloc[:,-1].values


X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 0 )


print(X_train)


print(X_test)


print(Y_train)


print(Y_test)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
import tensorflow as tf

import keras  as ke

from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(units=16,kernel_initializer='uniform',activation='relu',input_dim = 178))


classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))


classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))


classifier.add(Dense(units = 1,kernel_initializer   = 'uniform', activation = 'sigmoid'))


classifier.compile(optimizer = 'ftrl', loss = 'binary_crossentropy', metrics = ['accuracy'])

classifier.fit(X_train, Y_train, batch_size = 10, epochs = 100)

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

y_pred

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test,y_pred)
```

ADAM OPTIMIZATION

```
from google.colab import drive

drive.mount('/content/gdrive')
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split


d1=pd.read_csv('/content/gdrive/MyDrive/dataset-org.csv',header=None)
print(d1)


X=pd.DataFrame(d1.iloc[:,:-1].values)
Y=d1.iloc[:,-1].values


X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 0 )


print(X_train)


print(X_test)


print(Y_train)


print(Y_test)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

import tensorflow as tf

import keras  as ke

from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(units=16,kernel_initializer='uniform',activation='relu',input_dim = 178))


classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))
```

```
classifier.add(Dense(units = 16,kernel_initializer = 'uniform', activation = 'relu'))

classifier.add(Dense(units = 1,kernel_initializer   = 'uniform', activation = 'sigmoid'))

classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

classifier.fit(X_train, Y_train, batch_size = 10, epochs = 100)

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

y_pred

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test,y_pred)
```