IMPORTING LIBRARIES

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import re
import datetime


from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.externals import joblib
from sklearn import svm
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier


import warnings
warnings.filterwarnings('ignore')
```

READING THE DATASET

```
kickstarter_data = pd.read_csv("Kickstarter.csv", index_col = 0)


kickstarter_data.drop(kickstarter_data[kickstarter_data.state == 'undefined'].index,in
kickstarter_data.drop(kickstarter_data[kickstarter_data.state == 'live'].index,inplace
kickstarter_data.drop(kickstarter_data[kickstarter_data.state == 'suspended'].index,in
kickstarter_data['state'] = kickstarter_data['state'].map({'failed' : 0,'canceled' : 0


#Convert into datetime format
col_date=['state_changed_at','created_at','launched_at','deadline']

for i in col_date:
    kickstarter_data[i] =  pd.to_datetime(kickstarter_data[i])


#Length of the description
kickstarter_data["desc_len"] = kickstarter_data["blurb"].map(lambda x: len(x.split(" "


#how early project was launched, created before deadline and status changed after dead
kickstarter_data['launched_b_deadline']=(kickstarter_data["deadline"]- kickstarter_dat
kickstarter_data['created_b_deadline']=(kickstarter_data["deadline"]- kickstarter_data
kickstarter_data['state_a_deadline']=(kickstarter_data["state_changed_at"]- kickstarte
```

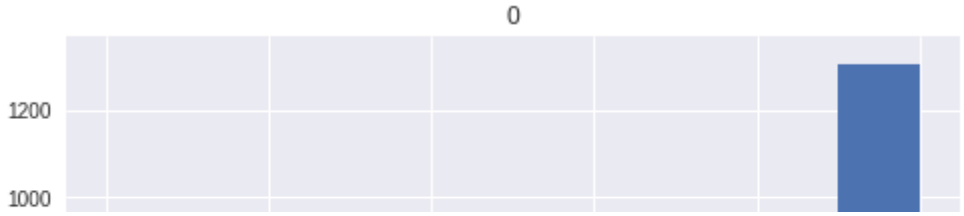
```
ignored_features =['blurb',
```

```
    'category',
    'converted_pledged_amount',
    'creator'                   ,
    'currency'                   ,
    'currency_symbol'             ,
    'currency_trailing_code'     ,
    'current_currency'            ,
    'disable_communication'       ,
    'friends'                     ,
    'fx_rate'                    ,
     'country',
    'id'                          ,
    'is_backing'                   ,
    'is_starrable'                 ,
    'is_starred'                   ,
    'location'                     ,
    'name'                         ,
    'permissions'                  ,
    'photo'                        ,
    'pledged'                      ,
    'profile'                      ,
    'slug'                         ,
    'source_url'   ,
    'spotlight'     ,
    'staff_pick'     ,
    'state'           ,
    'static_usd_rate'              ,
    'urls'                         ,
    'usd_pledged'                  ,
    'usd_type'   ]  +col_date


    features = list(kickstarter_data.columns)
    features = [i for i in features if i not in ignore_features]
    label = "state"


    X = kickstarter_data[features].values
    y = kickstarter_data[label].values
```

Checking that your target variable is binary

```
    sample_y = pd.DataFrame(y)


    sample_y.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7ffa257e8390>]],
      dtype=object)
```



Step 1: DATA PREPROCESSING

```
#Encoding the categorical features(the independent variables)

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder_X_1 = LabelEncoder()
X[:,1] = labelencoder_X_1.fit_transform(X[:,1])

labelencoder_X_2 = LabelEncoder()
X[:,2] = labelencoder_X_2.fit_transform(X[:,2])

labelencoder_X_3 = LabelEncoder()
X[:,3] = labelencoder_X_3.fit_transform(X[:,3])


onehotencoder = OneHotEncoder(categorical_features = [1,2,3])
X = onehotencoder.fit_transform(X).toarray()


#splitting the dataset into training and testing dataset

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=


print(X_train.shape, X_test.shape)
```

```
(1156, 1845) (289, 1845)
```

```
#feature scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Step 2: CREATING AN ANN

```
!pip install --ignore-installed --upgrade https://storage.googleapis.com/tensorflow/wi
import warnings
warnings.filterwarnings('ignore')
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

```
from keras import optimizers
```

```
tensorflow-1.0.1-cp35-cp35m-win_intel64.whl is not a supported wheel on this pl
```

```
#Initaializing the ANN
classifier = Sequential()

#Adding the input layer and the first hidden layer with dropout
classifier.add(Dense(activation="relu", kernel_initializer="uniform", input_dim=X_trai
classifier.add(Dropout(rate = 0.2))

#Adding the second hidden layer
classifier.add(Dense(activation="relu", kernel_initializer="uniform", units=X_train.sh
classifier.add(Dropout(rate = 0.2))

#Adding the output layer
classifier.add(Dense(activation="sigmoid", kernel_initializer="uniform", units=1))

#Compiling the ANN
classifier.compile(optimizer = 'adam', loss = "binary_crossentropy", metrics=["accurac
```

```
#Fitting the ANN to training set
classifier.fit(X_train, y_train, batch_size=50, epochs=50)
```

```
1156/1156 [==============================] - 0s 294us/step - loss: 4.9692e-05 -
Epoch 23/50
1156/1156 [==============================] - 0s 271us/step - loss: 2.9978e-05 -
Epoch 24/50
1156/1156 [==============================] - 0s 290us/step - loss: 3.1572e-05 -
Epoch 25/50
1156/1156 [==============================] - 0s 254us/step - loss: 3.0011e-05 -
Epoch 26/50
1156/1156 [==============================] - 0s 271us/step - loss: 4.3865e-05 -
Epoch 27/50
1156/1156 [==============================] - 0s 281us/step - loss: 2.4209e-05 -
Epoch 28/50
1156/1156 [==============================] - 0s 244us/step - loss: 2.5257e-05 -
Epoch 29/50
1156/1156 [==============================] - 0s 256us/step - loss: 1.9940e-05 -
Epoch 30/50
1156/1156 [==============================] - 0s 268us/step - loss: 1.5944e-05 -
Epoch 31/50
1156/1156 [==============================] - 0s 268us/step - loss: 2.1084e-05 -
Epoch 32/50
1156/1156 [==============================] - 0s 249us/step - loss: 2.4073e-05 -
Epoch 33/50
1156/1156 [==============================] - 0s 253us/step - loss: 2.0549e-05 -
Epoch 34/50
1156/1156 [==============================] - 0s 263us/step - loss: 3.0065e-05 -
```

Step 3 : MAKING THE PREDICTIONS

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

Step 4 : EVALUATE THE MODEL

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_prediction, target_names=['0','1']))
```

```
              precision    recall  f1-score   support

           0       0.41      0.22      0.29        32
           1       0.91      0.96      0.93       257

   micro avg       0.88      0.88      0.88       289
   macro avg       0.66      0.59      0.61       289
weighted avg       0.85      0.88      0.86       289
```

```
1156/1156 [==============================] - 0s 280us/step - loss: 5.7648e-06 -
Epoch 50/50
1156/1156 [==============================] - 0s 292us/step - loss: 1.0629e-05 -
<keras.callbacks.History at 0x7ffa26ba9358>
```

```
<keras.callbacks.History at 0x7f1a26ba9358>
```