```
import pickle
import pandas as pd
import numpy as np
from sklearn.metrics import f1_score
import time
from tensorflow.keras.models import load_model
```

```
%tensorflow_version 2.x
```

```
import tensorflow
print(tensorflow.__version__)
```

```
2.5.0
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
file = "/content/drive/MyDrive/Pickle_files/median_values.pkl"
with open(file,'rb') as file:
  median_values = pickle.load(file)
```

```
file = "/content/drive/MyDrive/Pickle_files/truncated_SVD.pkl"
with open(file,'rb') as file:
  Trunc_SVD = pickle.load(file)
```

```
file = "/content/drive/MyDrive/Pickle_files/MinMaxSc.pkl"
with open(file,'rb') as file:
  MinMaxSc = pickle.load(file)
```

```
file = "/content/drive/MyDrive/Pickle_files/clf_inv.pkl"
with open(file,'rb') as file:
  clf_inv = pickle.load(file)
```

```
file = "/content/drive/MyDrive/rf_best.pkl"
with open(file,'rb') as file:
  rf_best = pickle.load(file)
```

```
load_encoder_model = load_model('/content/drive/MyDrive/Pickle_files/encoder.h5')
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was
```

```
file = "/content/drive/MyDrive/Pickle_files/clf_inv.pkl"
with open(file,'rb') as file:
  clf_best = pickle.load(file)
```

```
  clf_best = pickle.load(file)
```

```
df = pd.read_csv("/content/drive/MyDrive/Kaggle_Training_Dataset_v2.csv")
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWa
  interactivity=interactivity, compiler=compiler, result=result)
```

```
df.head()
```

	sku	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_mo
0	1026827	0.0	NaN	0.0	0.0	
1	1043384	2.0	9.0	0.0	0.0	
2	1043696	2.0	NaN	0.0	0.0	
3	1043852	7.0	8.0	0.0	0.0	
4	1044048	8.0	NaN	0.0	0.0	

```
def final_fun_1(X):
  #These Two columns contains more than 95% of 0s
  zero_columns = ['pieces_past_due','local_bo_qty']
  X = X.drop(columns=zero_columns,axis=1)

  # replacing -99 by Nan in performance column
  X.perf_6_month_avg.replace({-99.0 : np.nan},inplace=True)
  X.perf_12_month_avg.replace({-99.0 : np.nan},inplace=True)

  # Converting categories like Yes and No to 0s and 1s
  categorical_columns = ['rev_stop','stop_auto_buy','ppap_risk','oe_constraint','deck
  for col in categorical_columns:
    X[col].replace({'Yes':1,'No':0},inplace=True)
    X[col]=X[col].astype(int)

  # Removing outliers points by taking only values below 99 percentile
  X = X[(df.national_inv >= 0.000) & (X.national_inv <= 5487.000) & (X.in_transit_qty
      (X.forecast_3_month <= 2280.000) & (X.forecast_6_month <= 4335.659999999916) &\
      (X.forecast_9_month <= 6316.000) & (X.sales_1_month <= 693.000) & (X.sales_3_mo
      (X.sales_6_month <= 4410.000) & (X.sales_9_month <= 6698.000) & (X.min_bank <=

  # Median Imputation
  X = X.fillna(median_values)

  # Getting SVD Features
  X_svd = Trunc_SVD.transform(X)

  # Encoder Model
  encoder_X = load_encoder_model.predict(X)

  # Dicretisation using Decision Tree
  X['national_inv_prob'] = clf_inv.predict_proba(X.national_inv.to_frame())[:,1]
```

```
  # creating bins for national_inv features
  X['national_inv_bins'] = 0
  X.loc[(X['national_inv'] == 0.0),'national_inv_bins'] = 1
  X.loc[(X['national_inv'] == 1.0),'national_inv_bins'] = 2
  X.loc[(X['national_inv'] >= 2.0) & (X['national_inv'] <= 9.0),'national_inv_bins']
  X.loc[(X['national_inv'] >= 10.0),'national_inv_bins'] = 4

  # Adding SVD and Encoder features in the main dataframe
  for i in range(2):
    X['T_SVD_'+str(i)] = X_svd[:,i]
    X['AutoEncoder_'+str(i)] = encoder_X[:,i]

  cols = X.columns

  # Performing MinMaxScaler on Data
  X = pd.DataFrame(MinMaxSc.transform(X),columns = cols)

  output = rf_best.predict(X)

  return output
```

```
data_temp = df.head()
target_data = df['went_on_backorder']
data_temp = data_temp.drop(['sku','went_on_backorder'],axis=1)

start_time = time.time()
output = final_fun_1(data_temp)
print("Output : ",output)
print("Time Taken for execution is {}".format((time.time() - start_time)))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: UserWarning: Boolear
Output :  [0 0 0 0 0]
Time Taken for execution is 3.7127180099487305
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 361 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 500 out of 500 | elapsed:    0.1s finished
```

```
def final_fun_2(X,Y):
  #These Two columns contains more than 95% of 0s
  zero_columns = ['pieces_past_due','local_bo_qty']
  X = X.drop(columns=zero_columns,axis=1)

  # replacing -99 by Nan in performance column
  X.perf_6_month_avg.replace({-99.0 : np.nan},inplace=True)
  X.perf_12_month_avg.replace({-99.0 : np.nan},inplace=True)

  # Converting the Target variable
  Y.replace({'Yes':1,'No':0},inplace=True)
  Y.astype(int)
```

```
  # Converting categories like Yes and No to 0s and 1s
  categorical_columns = ['rev_stop','stop_auto_buy','ppap_risk','oe_constraint','deck
  for col in categorical_columns:
    X[col].replace({'Yes':1,'No':0},inplace=True)
    X[col]=X[col].astype(int)

  # Appending Target Variable back to dataframe so that we can remove the Outlier row
  X['went_on_backorder'] = Y

  # Removing outliers points by taking only values below 99 percentile
  X = X[(X.national_inv >= 0.000) & (X.national_inv <= 5487.000) & (X.in_transit_qty
      (X.forecast_3_month <= 2280.000) & (X.forecast_6_month <= 4335.659999999916) &\
      (X.forecast_9_month <= 6316.000) & (X.sales_1_month <= 693.000) & (X.sales_3_mo
      (X.sales_6_month <= 4410.000) & (X.sales_9_month <= 6698.000) & (X.min_bank <=

  print("Shape of outlier free dataframe :",X.shape)

  # Assign the Outlier free Target variable back to Y and Drop the Target variable fr
  Y = X['went_on_backorder']
  X = X.drop(columns='went_on_backorder',axis=1)

  # Median Imputation
  X = X.fillna(median_values)

  # Getting SVD Features
  X_svd = Trunc_SVD.transform(X)

  # Encoder Model
  encoder_X = load_encoder_model.predict(X)

  # Dicretisation using Decision Tree
  X['national_inv_prob'] = clf_inv.predict_proba(X.national_inv.to_frame())[:,1]

  # creating bins for national_inv features
  X['national_inv_bins'] = 0
  X.loc[(X['national_inv'] == 0.0),'national_inv_bins'] = 1
  X.loc[(X['national_inv'] == 1.0),'national_inv_bins'] = 2
  X.loc[(X['national_inv'] >= 2.0) & (X['national_inv'] <= 9.0),'national_inv_bins']
  X.loc[(X['national_inv'] >= 10.0),'national_inv_bins'] = 4

  # Adding SVD and Encoder features in the main dataframe
  for i in range(2):
    X['T_SVD_'+str(i)] = X_svd[:,i]
    X['AutoEncoder_'+str(i)] = encoder_X[:,i]

  cols = X.columns

  # Performing MinMaxScaler on Data
  X = pd.DataFrame(MinMaxSc.transform(X),columns = cols)

  output = rf_best.predict(X)
  f1 = f1_score(Y,output,average="macro")

  return f1
```

```
1  data_temp = df.head(1500)
2  target_data = data_temp['went_on_backorder']
3  data_temp = data_temp.drop(['sku','went_on_backorder'],axis=1)
```

```
Data Temp shape :  (1500, 21)
Target Data shape :  (1500,)
```

```
1  start_time = time.time()
2  output = final_fun_2(data_temp,target_data)
3  print("F1_Score : ",output)
4  print("Time Taken for execution is {}".format((time.time() - start_time)))
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4582: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  method=method,
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.1s
Shape of outlier free dataframe : (1466, 20)
F1_Score :  0.6994875298940895
Time Taken for execution is 0.4376845359802246
[Parallel(n_jobs=2)]: Done 361 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 500 out of 500 | elapsed:    0.2s finished
```