Packages

```
In [ ]: !pip -q install accelerate -U
        !pip -q install transformers[torch]
        !pip -q install datasets
        #Restart after installing
```

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly.express as px
        from transformers import pipeline
```

```
In [ ]: # Textwrp function to display the output in a better format
        # This is an optional function, you can ignore it
        from IPython.display import HTML, display

        def wrap_display():
          display(HTML('''
          <style>
            pre {
                white-space: pre-wrap;
            }
          </style>
          '''))
        get_ipython().events.register('pre_run_cell', wrap_display)
```

Bank Complaints Data

```
In [ ]: !wget https://github.com/venkatareddykonasani/Datasets/raw/master/Bank_Custome
        !unzip -o complaints_v2.zip
        complaints_data = pd.read_csv("/content/complaints_v2.csv")
        complaints_data.head()
```

Use distilbert model without finetunung

```
In [ ]: # Distil bert model
        from transformers import pipeline
        distilbert_model = pipeline(task="text-classification",
                                    model="distilbert-base-uncased",
                                    )
```

```
In [ ]: sample_data=complaints_data.sample(100, random_state=42)
        sample_data["text"]=sample_data["text"].apply(lambda x: " ".join(x.split()[:35
        sample_data["bert_predicted"] = sample_data["text"].apply(lambda x: distilbert
        #Default prediction is not a number LABEL_1, LABEL_0
        sample_data["bert_predicted_num"]=sample_data["bert_predicted"].apply(lambda x
        sample_data["bert_predicted_num"] = sample_data["bert_predicted_num"].astype(i
sample_data.head()
```

Accuracy of the model without fine-tuning

```
In [ ]: from sklearn.metrics import confusion_matrix
        cm = confusion_matrix(sample_data["label"], sample_data["bert_predicted_num"])
        print(cm)
        accuracy=cm.diagonal().sum()/cm.sum()
        print(accuracy)
```

Project - Finetuning the model with our data

```
In [ ]: !pip -q install accelerate -U
        !pip -q install transformers[torch]
        !pip -q install datasets
```

```
In [ ]: from transformers import DistilBertTokenizer, DistilBertForSequenceClassificat
        from transformers import Trainer, TrainingArguments
        from datasets import load_dataset, DatasetDict, ClassLabel, Dataset
        import pandas as pd
        from sklearn.model_selection import train_test_split
        import torch
```

```
In [ ]: #The target variable must be named as "label" - Verify it, before proceeding
        print(sample_data.columns)
```

```
In [ ]: Sample_data = Dataset.from_pandas(sample_data)
        # Split the dataset into training and testing sets
        train_test_split = Sample_data.train_test_split(test_size=0.2)  # 80% training
        dataset = DatasetDict({
            'train': train_test_split['train'],
            'test': train_test_split['test']
        })
        dataset
```

Load the tokenizer

```
In [ ]: # Load the tokenizer
        tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

        # Padding
        tokenizer.pad_token = tokenizer.eos_token
        tokenizer.pad_token_id = tokenizer.eos_token_id
        tokenizer.add_special_tokens({'pad_token': '[PAD]'} )

        def tokenize_function(examples):
            return tokenizer(examples["text"], padding="max_length", truncation=True,
        tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

Load and Train the model

```
In [ ]: model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-u
                                                                    num_labels=2,
                                                                    pad_token_id=toker
        model
```

```
In [ ]: training_args = TrainingArguments(
            output_dir="./results_bert_custom",
            num_train_epochs=1,
            logging_dir="./logs_bert_custom",
            evaluation_strategy="epoch",
        )

        # Initialize the Trainer
        trainer = Trainer(
            model=model,
            args=training_args,
            train_dataset=tokenized_datasets['train'],
            eval_dataset=tokenized_datasets['test'],
        )

        # Start training
        trainer.train()
```

```
In [ ]: # Define the directory where you want to save your model and tokenizer
        model_dir = "./distilbert_finetuned"

        # Save the model
        model.save_pretrained(model_dir)

        # Save the tokenizer
        tokenizer.save_pretrained(model_dir)

        #Save the model with
        trainer.save_model('Distilbert_CustomModel_10K')
```

```
In [ ]: def make_prediction(text):
          new_complaint=text
          inputs=tokenizer(new_complaint, return_tensors="pt")
          inputs = inputs.to(torch.device("cuda:0"))
          outputs=model(**inputs)
          predictions=outputs.logits.argmax(-1)
          predictions=predictions.detach().cpu().numpy()
          return(predictions)

        sample_data["finetuned_predicted"]=sample_data["text"].apply(lambda x: make_pr
        sample_data.sample(10)
```

```
In [ ]: from sklearn.metrics import confusion_matrix
        # Create the confusion matrix
```

```
cm1 = confusion_matrix(sample_data["label"], sample_data["finetuned_predicted"
print(cm1)
accuracy1=cm1.diagonal().sum()/cm1.sum()
print(accuracy1)
```

Loading a pre-built model and making prediction

```
In [ ]: #Code to donwloading the distilbert model
        !gdown --id 1785J3ir19RaZP3ebbFvWUX88PMaBouro -O distilbert_finetuned_V1.zip
        !unzip -o -j distilbert_finetuned_V1.zip -d distilbert_finetuned_V1

        model_v1 = DistilBertForSequenceClassification.from_pretrained('/content/disti
        model_v1.to("cuda:0")
```

```
In [ ]: def make_prediction(text):
          new_complaint=text
          inputs=tokenizer(new_complaint, return_tensors="pt")
          inputs = inputs.to(torch.device("cuda:0"))
          outputs=model_v1(**inputs)
          predictions=outputs.logits.argmax(-1)
          predictions=predictions.detach().cpu().numpy()
          return(predictions)
```

```
In [ ]: sample_data_large=complaints_data.sample(n=1000, random_state=55)
        sample_data_large["finetuned_predicted"]=sample_data_large["text"].apply(lambd
```

```
In [ ]: sample_data_large["finetuned_predicted"]
```

```
In [ ]: from sklearn.metrics import confusion_matrix
        # Create the confusion matrix
        cm1 = confusion_matrix(sample_data_large["label"], sample_data_large["finetune
        print(cm1)
        accuracy1=cm1.diagonal().sum()/cm1.sum()
        print(accuracy1)
```

Saving the Model on HuggingFace hub

```
In [ ]: !pip install transformers
        !pip install huggingface_hub
        !pip install -U ipykernel #for executing the commands
```

```
In [ ]: from transformers import DistilBertTokenizer, DistilBertForSequenceClassificat
```

```
In [ ]: !gdown --id 1785J3ir19RaZP3ebbFvWUX88PMaBouro -O distilbert_finetuned_V1.zip
        !unzip -o -j distilbert_finetuned_V1.zip -d distilbert_finetuned_V1

        model = DistilBertForSequenceClassification.from_pretrained('/content/distilbe
```

In [ ]:
```
import os
os.environ['HUGGINGFACEHUB_API_TOKEN']="YOUR ACCESS TOKEN"
```

In [ ]:
```
from huggingface_hub import notebook_login
notebook_login()
#To get Auth token: Profile >> Settings >>Access Token
```

In [ ]:
```
model.push_to_hub("pratik456ailab/Bank_distil_bert_10K")
```

Loading the model from HuggingFace hub

In [ ]:
```
model=DistilBertForSequenceClassification.from_pretrained("pratik456ailab/Bank
```

In [ ]:
```
from transformers import DistilBertTokenizer, DistilBertForSequenceClassificat

tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

In [ ]:
```
import pandas as pd
!wget https://github.com/venkatareddykonasani/Datasets/raw/master/Bank_Custome
!unzip -o complaints_v2.zip
complaints_data = pd.read_csv("/content/complaints_v2.csv")
list(complaints_data["text"].head())
```

In [ ]:
```
import torch
```

In [ ]:
```
complaint="""
payment history missing credit report made mistake put account forbearance wit
"""

inputs=tokenizer(complaint, return_tensors="pt")
outputs=model(**inputs)
predictions=outputs.logits.argmax(-1)
predictions=predictions.detach().cpu().numpy()
print(predictions)
```

Web App Creation

In [ ]:
```
%%writefile requirements.txt
streamlit
numpy
pandas
torch
transformers
huggingface_hub
```

In [ ]:
```
!pip install -r requirements.txt
```

In [ ]:
```
%%writefile app.py
import streamlit as st
import numpy as np
import pandas as pd
import torch
from transformers import DistilBertTokenizer, DistilBertForSequenceClassificat

tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
model = DistilBertForSequenceClassification.from_pretrained('pratik456ailab/Ba

st.title("Bank Complaints Categorization")
st.write("Sample Complaints are given below")
Sample_Complaints = [
    {"Sentence": "Credit Report - payment history missing credit report made m
    {"Sentence": "Retail Related - forwarded message cc sent friday pdt subjec
]
st.table(Sample_Complaints)
user_input = st.text_input("Enter a complaint:")
button=st.button("Classify")

d={
    0: "Credit reporting",
    1: "Mortgage and Others"
}

if user_input and button:
  inputs=tokenizer(user_input, return_tensors="pt")
  outputs=model(**inputs)
  predictions=outputs.logits.argmax(-1)
  predictions=predictions.detach().cpu().numpy()
  print(predictions)
  st.write("Prediction :" , d[predictions[0]])
```

In [ ]:
```
!streamlit run app.py & npx localtunnel --port 8501 & curl ipv4.icanhazip.com

#This sometimes doesn't work on Chrome
```

In [ ]:
```

```