```
import pandas as pd
```

```
df = pd.read_excel("DM.xlsx")
```

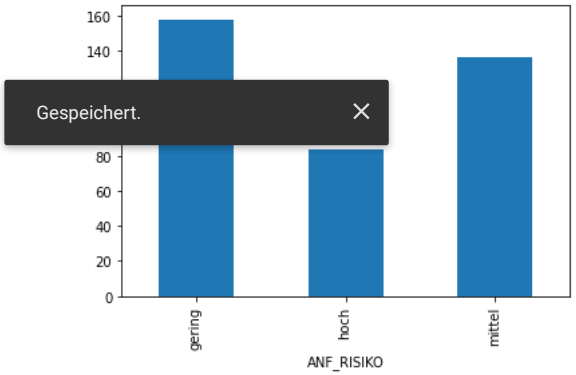
```
/usr/local/lib/python3.8/dist-packages/openpyxl/styles/stylesheet.py:226: UserWarning: Workbook contains no default sty
  warn("Workbook contains no default style, apply openpyxl's default")
```

```
df = df[["ANF_BESCHREIBUNG", "ANF_RISIKO"]]
#df['ANF_RISIKO'] = df['ANF_RISIKO'].replace("gering", 3)
#df['ANF_RISIKO'] = df['ANF_RISIKO'].replace("mittel", 2)
#df['ANF_RISIKO'] = df['ANF_RISIKO'].replace("hoch", 1)
df.head()
```

| | ANF_BESCHREIBUNG | ANF_RISIKO |
|---|---|---|
| 0 | Nach Reindizierung der Indexklasse wird der Ei... | mittel |
| 1 | Nach Reindizierung der Indexklasse wird der Ei... | mittel |
| 2 | Sollte es nur einen Treffer geben, muss dieser... | gering |
| 3 | Kopieren aus einer Indexklasse und einfügen in... | gering |
| 4 | Es sind die inneren Rahmen gemeint | gering |

```
df.groupby(['ANF_RISIKO']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe978dda850>
```




Gespeichert.

```
pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.25.1-py3-none-any.whl (5.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.8/5.8 MB 54.6 MB/s eta 0:00:00
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from transformers) (21.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (1.21.6)
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from transformers) (3.9.0)
```

Ressourcen ×

Sie haben kein Abo. Weitere Informationen.
Derzeit sind keine Recheneinheiten verfügbar. Die Verfügbarkeit kostenloser Ressourcen wird nicht garantiert. Hier können Sie weitere Einheiten erwerben.
Sitzungen verwalten

Sie möchten mehr Arbeitsspeicher und Speicherplatz?
Upgrade auf Colab Pro ausführen

des Google Compute Engine-Back-Ends in Python 3
Ressourcen werden seit 21:01 angezeigt

System-RAM Laufwerk

```
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from transformers) (2.25.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.8/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from transformers) (6.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 105.7 MB/s eta 0:00:00
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (2022.6.
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 182.4/182.4 KB 28.3 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-h
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist-packages (from packaging>=20.0
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->transfor
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->transformers
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->transformer
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.11.1 tokenizers-0.13.2 transformers-4.25.1
```

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-german-cased')

example_text = 'Ich werde heute lange schlafen'
bert_input = tokenizer(example_text,padding='max_length', max_length = 10,
                       truncation=True, return_tensors="pt")


print(bert_input['input_ids'])
print(bert_input['token_type_ids'])
print(bert_input['attention_mask'])
```

```
Downloading: 100%                                   255k/255k [00:00<00:00, 1.73MB/s]

                                                    29.0/29.0 [00:00<00:00, 896B/s]

                                                    433/433 [00:00<00:00, 7.41kB/s]
tensor([[    3,  1671,  1631,  1138,  2197, 21872,     4,     0,     0,     0]])
tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
tensor([[1, 1, 1, 1, 1, 1, 1, 0, 0, 0]])
```

Gespeichert.

```
example_text = tokenizer.decode(bert_input.input_ids[0])

print(example_text)
```

```
[CLS] Ich werde heute lange schlafen [SEP] [PAD] [PAD] [PAD]
```

```
import torch
import numpy as np
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-german-cased')
labels = {'gering':3,
          'mittel':2,
          'hoch':1
```

```
          }

class Dataset(torch.utils.data.Dataset):

    def __init__(self, df):

        self.labels = [labels[label] for label in df['ANF_RISIKO']]
        self.texts = [tokenizer(text,
                                padding='max_length', max_length = 512, truncation=True,
                                 return_tensors="pt") for text in df['ANF_BESCHREIBUNG']]

    def classes(self):
        return self.labels

    def __len__(self):
        return len(self.labels)

    def get_batch_labels(self, idx):
        # Fetch a batch of labels
        return np.array(self.labels[idx])

    def get_batch_texts(self, idx):
        # Fetch a batch of inputs
        return self.texts[idx]

    def __getitem__(self, idx):

        batch_texts = self.get_batch_texts(idx)
        batch_y = self.get_batch_labels(idx)

        return batch_texts, batch_y
```

```
np.random.seed(1234)
df_train, df_val, df_test = np.split(df.sample(frac=1, random_state=42),
                                     [int(.8*len(df)), int(.9*len(df))])

print(len(df_train), len(df_val), len(df_test))
```

Gespeichert.

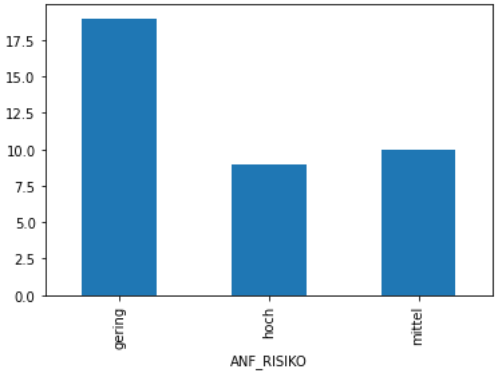
```
302 38 38
```

```
df_train.groupby(['ANF_RISIKO']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe89fb4cc10>
```

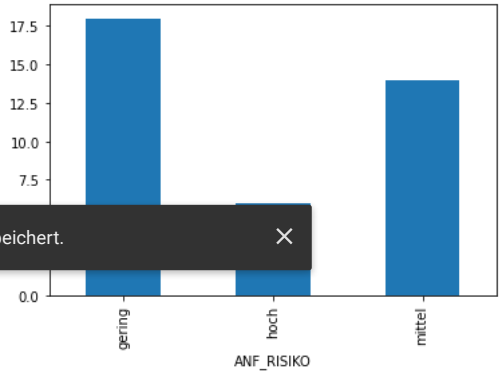

```
df_test.groupby(['ANF_RISIKO']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe917df4190>
```




```
df_val.groupby(['ANF_RISIKO']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe917dec0d0>
```




Gespeichert.

```
df_train
```

| | ANF_BESCHREIBUNG | ANF_RISIKO |
|---|---|---|
| **287** | Export ist auch mit Schadennummer = 0 möglich... | gering |
| **329** | Bei der Verarbeitung eines Dokuments der Index... | gering |
| **323** | Analog zur Funktion Seite ignorieren kann auch... | gering |
| **145** | INFORMATION\nAbhängig von der Auswahl im Feld ... | hoch |
| **55** | \r\nVORBEDINGUNG\r\n- Trennblatt Makro wurde i... | mittel |

```
from torch import nn
from transformers import BertModel

class BertClassifier(nn.Module):

    def __init__(self, dropout=0.5):

        super(BertClassifier, self).__init__()

        self.bert = BertModel.from_pretrained('bert-base-german-cased')
        self.dropout = nn.Dropout(dropout)
        self.linear = nn.Linear(768, 3)
        self.relu = nn.ReLU()

    def forward(self, input_id, mask):

        _, pooled_output = self.bert(input_ids= input_id, attention_mask=mask,return_dict=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        final_layer = self.relu(linear_output)

        return final_layer
```

Gespeichert.

```
def train(model, train_data, val_data, learning_rate, epochs):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=2, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    criterion = nn.CrossEntropyLoss()
    optimizer = Adam(model.parameters(), lr= learning_rate)

    if use_cuda:

            model = model.cuda()
            criterion = criterion.cuda()
```

```
    train_loss = []
    train_acc = []
    val_loss = []
    val_acc = []

    for epoch_num in range(epochs):

            total_acc_train = 0
            total_loss_train = 0

            for train_input, train_label in tqdm(train_dataloader):

                train_label = train_label.to(device)
                mask = train_input['attention_mask'].to(device)
                input_id = train_input['input_ids'].squeeze(1).to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, train_label.long())
                total_loss_train += batch_loss.item()

                acc = (output.argmax(dim=1) == train_label).sum().item()
                total_acc_train += acc

                model.zero_grad()
                batch_loss.backward()
                optimizer.step()

            total_acc_val = 0
            total_loss_val = 0

            with torch.no_grad():

                for val_input, val_label in val_dataloader:

                    val_label = val_label.to(device)
                    mask = val_input['attention_mask'].to(device)
                    input_id = val_input['input_ids'].squeeze(1).to(device)

                    output = model(input_id, mask)

                    batch_loss = criterion(output, val_label.long())
                    total_loss_val += batch_loss.item()

                    acc = (output.argmax(dim=1) == val_label).sum().item()
                    total_acc_val += acc


            train_loss = np.append(train_loss, (total_loss_train / len(train_data)))
            train_acc = np.append(train_acc, (total_acc_train / len(train_data)))
            val_loss = np.append(val_loss, (total_loss_val / len(val_data)))
            val_acc = np.append(val_acc, (total_acc_val / len(val_data)))

    return train_loss, train_acc, val_loss, val_acc
```

Gespeichert.

```
EPOCHS = 2
model = BertClassifier()
LR = 1e-5

loss_tr, acc_tr, loss_val, acc_val = train(model, df_train, df_val, LR, EPOCHS)
```

```
Some weights of the model checkpoint at bert-base-german-cased were not used when initializing BertModel: ['cls.predict
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with ano
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly i
100%|██████████| 151/151 [30:06<00:00, 11.96s/it]
100%|██████████| 151/151 [29:34<00:00, 11.75s/it]
```

```
print("loss_tr: ", loss_tr)
print("acc_tr: ", acc_tr)
print("loss_val: ", loss_val)
print("acc_val: ", acc_val)
```

```
loss_tr:  [0.59217591 0.30412015]
acc_tr:  [0.5397351 0.8013245]
loss_val:  [0.50499116 0.33362606]
acc_val:  [0.63157895 0.73684211]
```

```
def evaluate(model, test_data):

    test = Dataset(test_data)

    test_dataloader = torch.utils.data.DataLoader(test, batch_size=1)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:
```

Gespeichert.

```
    total_acc_test = 0
    zuhochkl = 0
    zuniedrigkl = 0
    richtigkl = 0
    with torch.no_grad():

        for test_input, test_label in test_dataloader:

              test_label = test_label.to(device)
              mask = test_input['attention_mask'].to(device)
              input_id = test_input['input_ids'].squeeze(1).to(device)

              output = model(input_id, mask)

              pred = output.argmax(dim=1)[0].item()
              trcl = test_label[0].item()

              if (pred < trcl):
                  zuhochkl = zuhochkl + 1
```

```
            if (pred > trcl):
                zuniedrigkl = zuniedrigkl + 1
            if (pred == trcl):
                richtigkl = richtigkl + 1

            acc = (output.argmax(dim=1) == test_label).sum().item()
            total_acc_test += acc

    print(f'Test Accuracy: {total_acc_test / len(test_data): .3f}')

    checksum = zuhochkl + zuniedrigkl + richtigkl
    print("zu hoch klassifiziert: ", zuhochkl)
    print("zu niedrig klassifiziert: ", zuniedrigkl)
    print("richtig klassifiziert: ", richtigkl)
    print("checksum: ", checksum)
    print("meine acc: ", richtigkl/checksum)
```

```
print(df_test.shape)
evaluate(model, df_test)
```

```
(38, 2)
Test Accuracy:  0.789
zu hoch klassifiziert:  1
zu niedrig klassifiziert:  7
richtig klassifiziert:  30
checksum:  38
meine acc:  0.7894736842105263
```

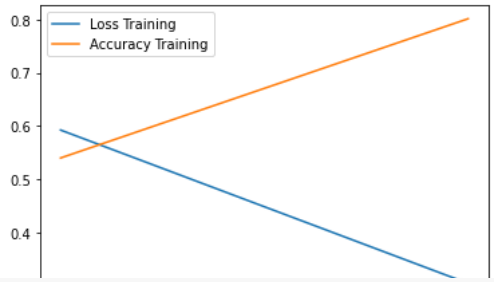
```
p1 = pd.DataFrame({
    'Loss Training': loss_tr,
    'Accuracy Training': acc_tr
    }, index=[1,2])

p2 = pd.DataFrame({
    'Loss Test': loss_val,
```

Gespeichert.

```
p1.plot.line()
p2.plot.line()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdef14c8f10>
```




```
def get_pred(model, test_data):

    test = Dataset(test_data)

    test_dataloader = torch.utils.data.DataLoader(test, batch_size=1)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:

        model = model.cuda()

    with torch.no_grad():

        pred = []

        for test_input, test_label in test_dataloader:

              test_label = test_label.to(device)
              mask = test_input['attention_mask'].to(device)
              input_id = test_input['input_ids'].squeeze(1).to(device)

              output = model(input_id, mask)

              if output.argmax(dim=1)[0].item() == 3:
                pred = np.append(pred, 'gering')
              if output.argmax(dim=1)[0].item() == 2:
                pred = np.append(pred, 'mittel')
              if output.argmax(dim=1)[0].item() == 1:
                pred = np.append(pred, 'hoch')

    test_data['Vorhersage'] = pred
    print(test_data)
```

Gespeichert.

```
var = pd.DataFrame({'ANF_BESCHREIBUNG': [
    "ich bin ein test text für das tolle modell",
    "ein text mit informationsdialog ist vielleicht richtig",
    "Die Sonne lacht vom Himmel doch die Software stürzt ab"
    ],
    'ANF_RISIKO': ["hoch", "gering", "mittel"]})
var.head()
```

| | ANF_BESCHREIBUNG | ANF_RISIKO |
|---|---|---|
| **0** | ich bin ein test text für das tolle modell | hoch |
| **1** | ein text mit informationsdialog ist vielleicht... | gering |
| **2** | Die Sonne lacht vom Himmel doch die Software s... | mittel |

```
get_pred(model, var)
```

```
                              ANF_BESCHREIBUNG ANF_RISIKO Vorhersage
0        ich bin ein test text für das tolle modell       hoch     gering
1  ein text mit informationsdialog ist vielleicht...     gering     gering
2  Die Sonne lacht vom Himmel doch die Software s...     mittel     gering
```

Gespeichert.

6 s Abgeschlossen um 22:06

Laufzeittyp ändern