```
import pandas as pd
```

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

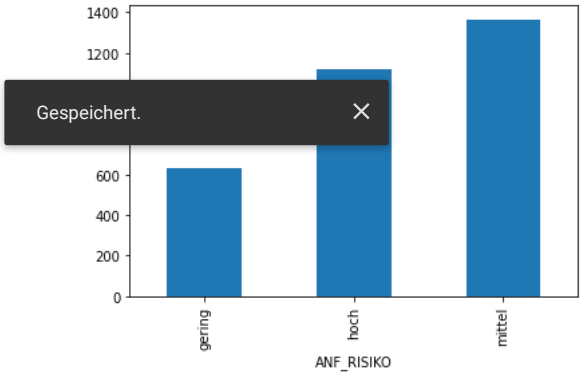
```
/usr/local/lib/python3.8/dist-packages/openpyxl/styles/stylesheet.py:226: UserWarning: Workbook contains no default sty
  warn("Workbook contains no default style, apply openpyxl's default")
```

```
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	CR 58564 Bankverb. mit dem Länderkennzeichen I...	mittel
1	CR 58564 Bankverb. mit dem Länderkennzeichen I...	mittel
2	CR55459 - Fachkonzept PTAR: 27075-EVS/K4\n\n- ...	mittel
3	CR55459 - Fachkonzept PTAR: 27075-EVS/K4\n\n- ...	mittel
4	Hier hat der Benutzer die Möglichkeit, Dokumen...	gering

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f138967f1c0>
```



Gespeichert.

```
pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.8/dist-packages (4.25.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.8/dist-packages (from transformer
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.8/dist-packages (from transf
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from transformers) (21.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from transformers) (3.9.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (1.21.6)
```

Ressourcen

Sie haben Colab Pro abonniert. Weitere Informationen.
Verfügbar: 98.69 Recheneinheiten
Nutzungsrate: ca. 1.96 pro Stunde
Sie haben 1 aktive Sitzung. Sitzungen verwalten

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

(GPU) des Google Compute Engine-Back-Ends in Python 3
Ressourcen werden seit 18:26 angezeigt

System-RAM GPU-RAM Laufwerk

```
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.8/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from transformers) (2.25.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (2022.6.
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: 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: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->transfor
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->transformer
```

```
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'])
```

```
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]])
```

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

print(example_text)
```

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

Gespeichert.

```
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
```

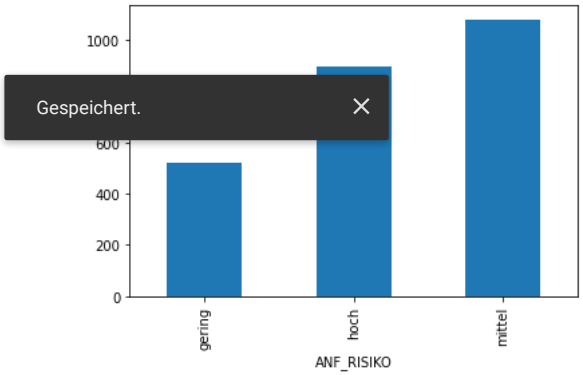
```
np.random.seed(1234)
df_train, df_val, df_test = np.split(df.sample(frac=1, random_state=42),
                                     [int(.8*len(df)), int(.9*len(df))])

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

```
2496 312 313
```

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

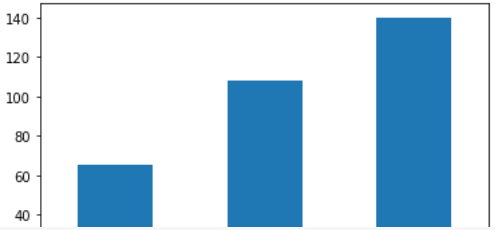
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f12b0c6b8e0>
```



Gespeichert.

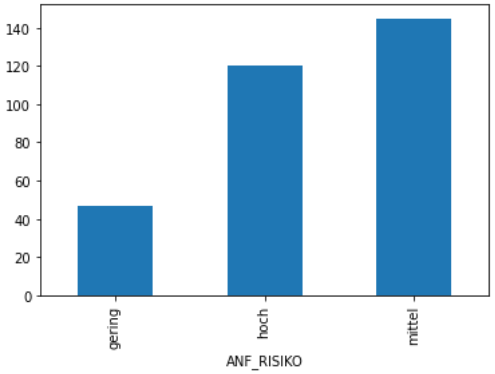
```
df_test.groupby(['ANF_RISIKO']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1328fabdc0>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1329380970>
```




```
df_train
```

	ANF_BESCHREIBUNG	ANF_RISIKO
	?Aplus: \nBasischutz:\n- Unfa...	gering
	256754004BDEED/106156F...	mittel
2131	REQM-345 Abbildung der Rahmenvereinbarungen St...	mittel
214	Die Partnergruppen Schnelleingabemaske wird ge...	gering
2948	Release 20.20: \nREQM-7041 PWN Betrieb&Beruf -...	gering
...	...	...
79	Ansichtsobjekte sind immer Read-Only-Daten zu ...	hoch
1172	mit Ursula Schreiner abgestimmtes Testset	hoch
814	Folgende Reiter sind als Ergebnis einer Abfrag...	mittel
1852	grundsätzlich gleich zu Kfz und Leasing mit fo...	gering
2220	Erstellung Testdaten für Alte Anträge Tests\n\...	hoch

2496 rows × 2 columns

Gespeichert.

```
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, 5)
        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
```

```
from torch.optim import Adam
from tqdm import tqdm

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()
                                                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):
```

Gespeichert.

```
            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.seq_rel
- 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%|██████████| 1248/1248 [04:37<00:00,  4.49it/s]
100%|██████████| 1248/1248 [04:43<00:00,  4.40it/s]
```

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

```
loss_tr:  [0.36199453 0.10969808]
acc_tr:  [0.68870192 0.92668269]
loss_val:  [0.18972439 0.11823913]
acc_val:  [0.85576923 0.91346154]
```

```
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:

        model = model.cuda()

    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)

              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)
```

Gespeichert.

```
    print("richtig klassifiziert: ", richtigkl)
    print("checksum: ", checksum)
    print("meine acc: ", richtigkl/checksum)
```

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

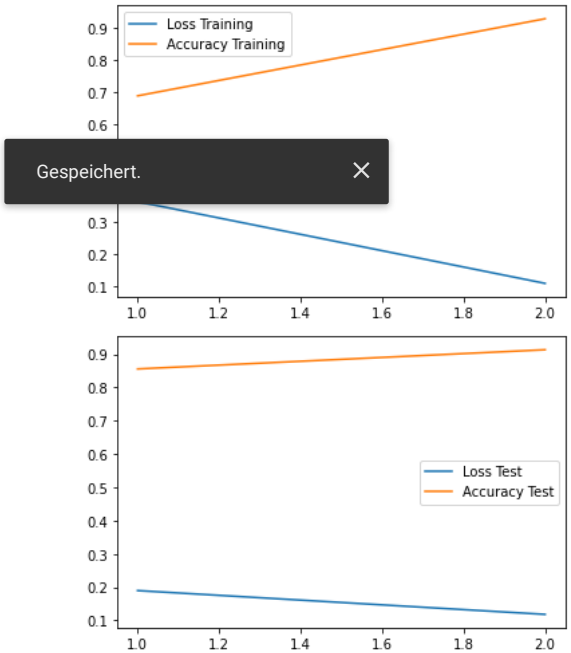
```
(313, 2)
Test Accuracy:  0.917
zu hoch klassifiziert:  13
zu niedrig klassifiziert:  13
richtig klassifiziert:  287
checksum:  313
meine acc:  0.9169329073482428
```

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

p2 = pd.DataFrame({
    'Loss Test': loss_val,
    'Accuracy Test': acc_val
    }, index=[1,2])

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1328f56160>
```



Gespeichert.

```
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.

```
    "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       hoch
```

```
1  ein text mit informationsdialog ist vielleicht...     gering       hoch
2  Die Sonne lacht vom Himmel doch die Software s...     mittel     mittel
```

Laufzeittyp ändern

0 s Abgeschlossen um 18:42

Gespeichert.