In [96]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from sys import platform
import numpy as np
```

In [97]:
```
df1 = pd.read_excel("Data/Web_Teaser_Bewertung_01.xlsx")#, sheet_name=None)
```

In [98]:
```
df2 = pd.read_excel("Data/Web_Teaser_Bewertung_02.xlsx")#, sheet_name=None)
```

In [99]:
```
df3 = pd.read_excel("Data/Web_Teaser_Bewertung_03.xlsx")#, sheet_name=None)
```

In [100]:
```
dfs = [df1, df2, df3]
```

In [101]:
```
target_names = ["accent colour", "background colour", "font colour", "font c
ontrast", "space", "roundness"]
```

In [102]:
```
def split_input_target(df):
    header = df.iloc[0]
    df = df.rename(columns=df.iloc[0]).iloc[1:201]
    if platform == "linux" or platform == "linux2":
        df = df.drop(columns="teaser_name")
    target_names = ["accent colour", "background colour", "font colour", "fo
nt contrast", "space", "roundness"]
    targets = df[target_names]
    inputs = df.drop(columns=target_names)
    inputs = inputs.astype(int)
    return inputs, targets
```

In [103]:
```
inputs = []
targets = []
for df in dfs:
    input_df, target_df = split_input_target(df)
    inputs.append(input_df)
    targets.append(target_df)
```

In [104]:
```
inputs1, targets1 = split_input_target(df1)
inputs2, targets2 = split_input_target(df2)
```

Check data integrity of target

In [105]:
```
(targets1 != targets2).sum()
```

Out[105]:
```
accent colour        53
background colour     1
font colour           0
font contrast         0
space                 0
roundness             8
dtype: int64
```

In [106]:
```
mask = (targets1["accent colour"] != targets2["accent colour"])
```

In [107]:
```
#targets1[mask]
```

In [108]:
```
#targets2[mask]
```

Show deviation of responses:

In [109]:
```
# Inpect data:
inputs[2]
```

Out[109]:

	Innovative - Traditional	Elegant - Lassig	Emotional - Sachlich	Jung - Erfahren	Perfektionistisch - Spontan	zurückhalten - selbstsicher	Weiblich - Männlich
1	-6	5	-10	-5	7	2	-8
2	1	2	2	4	2	5	6
3	-8	4	-6	-6	8	7	-6
4	-3	4	4	-2	3	-7	3
5	-5	3	-7	-7	6	-4	-4
...	...	...	...	...	...	...	...
196	6	-4	6	8	-6	-6	6
197	-5	6	-5	-2	8	8	6
198	3	6	7	6	3	-6	8
199	3	-3	6	6	-6	0	6
200	-3	-3	-3	3	6	0	8

200 rows × 7 columns

In [110]:
```
# Calculate mean
mean_input = inputs[0].copy()
for inp in inputs[1:]:
    mean_input += inp.copy()
mean_input /= len(inputs)
```

In [111]:
```
mean_input.mean()
```

Out[111]:
```
Innovative - Traditional      -0.803333
Elegant - Lassig               1.336667
Emotional - Sachlich          -1.005000
Jung - Erfahren                0.520000
Perfektionistisch - Spontan    1.938333
zurückhalten - selbstsicher   -4.108333
Weiblich -  Männlich           1.186667
dtype: float64
```

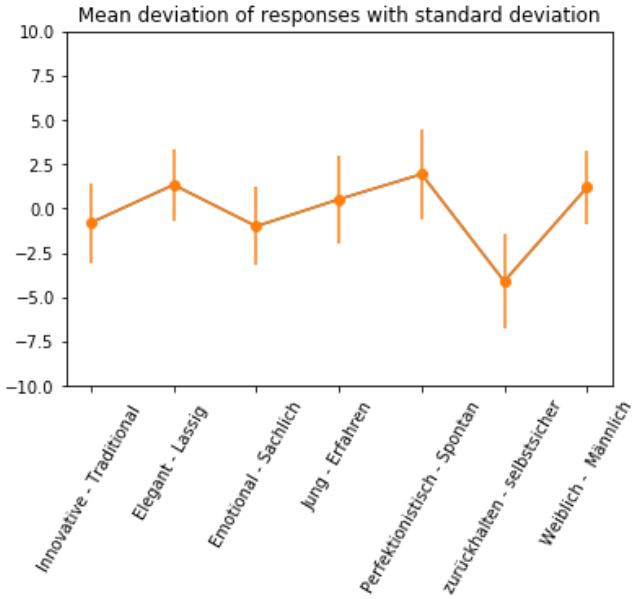
In [112]:
```
# Calculate std:
std_input = (inputs[0].copy() - mean_input) ** 2
for inp in inputs[1:]:
    std_input += (inp.copy() - mean_input) ** 2
std_input /= len(inputs)
std_input = std_input ** 0.5
```

In [113]:
```
std_input.mean()
```

Out[113]:
```
Innovative - Traditional       2.259430
Elegant - Lassig               1.998557
Emotional - Sachlich           2.192584
Jung - Erfahren                2.502932
Perfektionistisch - Spontan    2.502957
zurückhalten - selbstsicher    2.707655
Weiblich -  Männlich           2.069644
dtype: float64
```

In [114]:
```
plt.plot(mean_input.mean())
plt.errorbar(range(len(mean_input.mean())), mean_input.mean(),yerr=std_inpu
t.mean(), fmt='-o')
plt.ylim(-10, 10)
plt.xticks(rotation=60)
plt.title("Mean deviation of responses with standard deviation")
```

Out[114]:
```
Text(0.5, 1.0, 'Mean deviation of responses with standard deviation')
```



Correlation between input columns

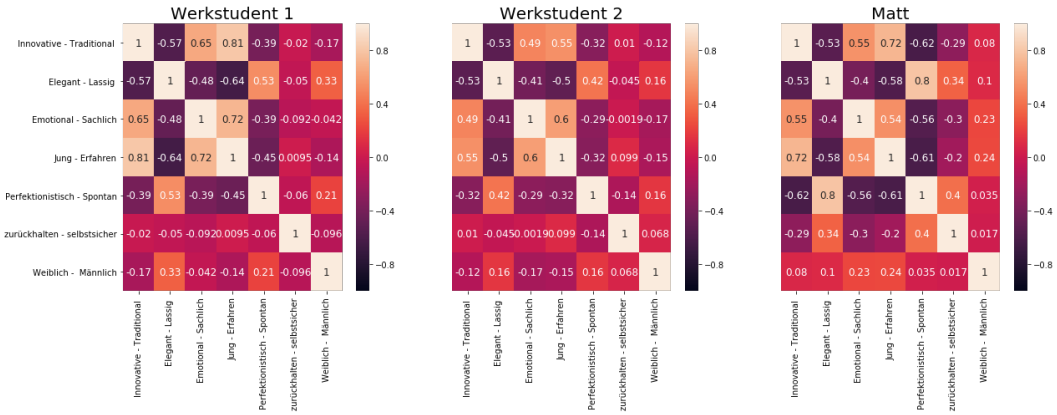
In [115]:
```
corr1 = inputs[0].corr()
corr2 = inputs[1].corr()
corr3 = inputs[2].corr()

corr_total = inputs[0].append(inputs[1]).append(inputs[2]).corr()
```

In [116]:
```
fig = plt.figure(figsize = (20, 20)) # width x height
ax1 = fig.add_subplot(331) # row, column, position
ax2 = fig.add_subplot(332)
ax3 = fig.add_subplot(333)

sns.heatmap(corr1, ax=ax1, vmin=-1, vmax=1,  annot=True, annot_kws={'fontsize': 12}).set_title('Werkstudent 1', fontsize =20)
sns.heatmap(corr2, ax=ax2, vmin=-1, vmax=1,  yticklabels=False,  annot=True, annot_kws={'fontsize': 12}).set_title('Werkstudent 2', fontsize =20)
sns.heatmap(corr3, ax=ax3, vmin=-1, vmax=1,  yticklabels=False, annot=True, annot_kws={'fontsize': 12}).set_title('Matt', fontsize =20)
```

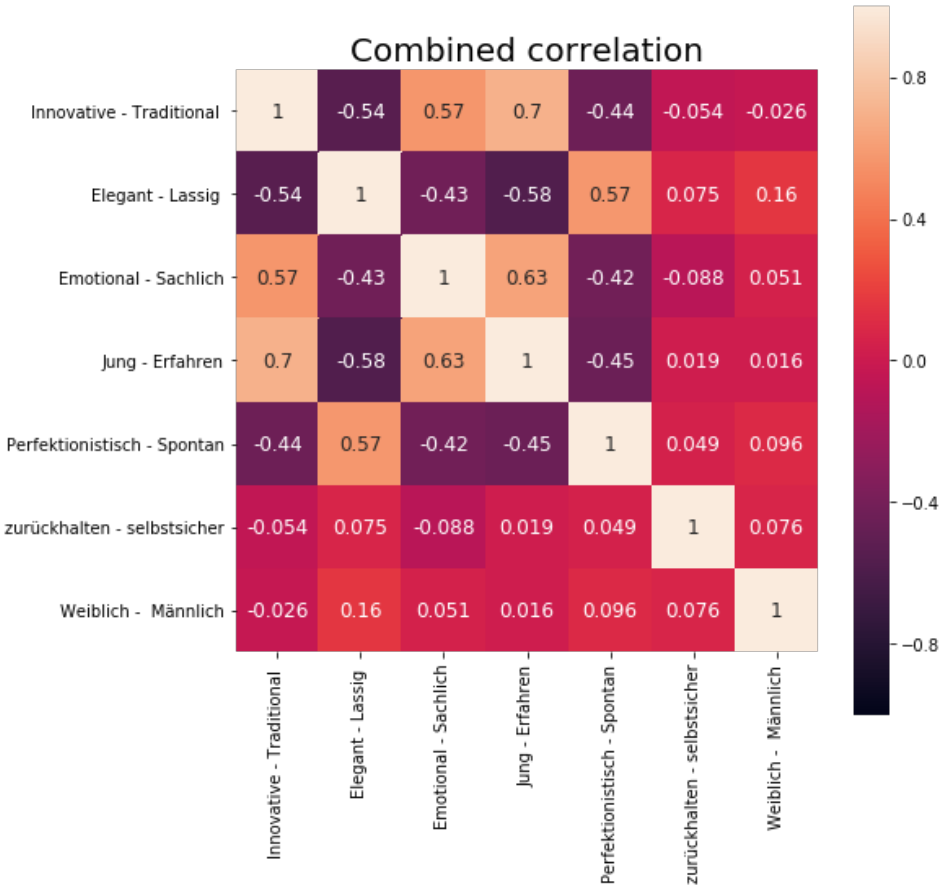
Out[116]: Text(0.5, 1.0, 'Matt')



In [117]:
```
fig = plt.figure(figsize = (8, 8)) # width x height
ax1 = fig.add_subplot(111)  # row, column, position

sns.heatmap(corr_total, ax=ax1, vmin=-1, vmax=1, square=True, annot=True, an
not_kws={'fontsize': 12}).set_title('Combined correlation', fontsize =20)
```

Out[117]: Text(0.5, 1.0, 'Combined correlation')



Sklearn tests:

In [118]:
```
targets[0]
```

Out[118]:

	accent colour	background colour	font colour	font contrast	space	roundness
1	#d24041	#ffffff	#1e2722	medium	tight	1
2	#7e0068	#ededed	#121212	medium	tight	1
3	#131313	#ffffff	#555555	low	medium	1
4	#e05e00	#fef9f4	#121212	low	tight	1
5	#c70200	#f7f6f7	#121212	medium	medium	1
...	...	...	...	...	...	...
196	#ff502d	#efecea	#000000	medium	medium	1
197	#9299a0	#ffffff	#2e3439	low	medium	1
198	#546276	#02234f	#ffc3bc	medium	medium	NaN
199	#005b55	#fdd6d0	#005c55	medium	medium	1
200	NaN	#000000	#ffffff	medium	tight	4

200 rows × 6 columns

In [119]:
```
# Check NANs
pd.concat(targets).isnull().mean()
```

Out[119]:
```
accent colour        0.265
background colour    0.005
font colour          0.000
font contrast        0.000
space                0.000
roundness            0.040
dtype: float64
```

In [120]:
```
dataset = pd.concat((pd.concat(inputs), pd.concat(targets)), axis=1)
```

In [121]:
```
dataset = dataset.dropna(subset=["roundness"])
```

In [122]:
```
inputs_np = dataset.drop(columns=target_names).to_numpy()
inputs_np.shape
```

Out[122]:
```
(576, 7)
```

In [123]:
```
# preprocess:
from sklearn.preprocessing import StandardScaler
inputs_np = StandardScaler().fit_transform(inputs_np)
```

In [124]:
```
pd.options.mode.chained_assignment = None  # default='warn''
```

In [125]:
```
# Extract target values that we want to predict:
used_targets = ["roundness"]
#used_targets = ["space", "font contrast", "roundness"]
targets = dataset[target_names]
targets["space"] = targets["space"].astype("category").cat.codes
targets["font contrast"] = targets["font contrast"].astype("category").cat.c
odes
targets["roundness"] = targets["roundness"].astype("category").cat.codes
targets_np = targets[used_targets].to_numpy()
```

In [126]:
```
from sklearn.model_selection import train_test_split
```

In [127]:
```
import sklearn
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
```

In [128]:
```
print(inputs_np.shape, targets_np.shape)
```

```
(576, 7) (576, 1)
```

In [129]:
```
def apply_classifier(inputs, targets, classifier):
    X_train, X_test, Y_train, Y_test = train_test_split(inputs_np, targets_n
p)
    classifier.fit(X_train, Y_train)
    Y_pred = classifier.predict(X_test)
    #print(sklearn.metrics.classification_report(Y_test, Y_pred))
    accuracy = classifier.score(X_test, Y_test)
    print("Accuracy: ", accuracy)
    return classifier
```

In [130]:
```
#classifier = SVC()
classifier = MultiOutputClassifier(SVC())
classifier = apply_classifier(inputs_np, targets_np, classifier)
```

```
Accuracy:  0.8402777777777778

/home/anton/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:193: Fu
tureWarning: The default value of gamma will change from 'auto' to 'scale' in
version 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [131]:
```
classifier = MultiOutputClassifier(MLPClassifier(solver='adam', max_iter=100
0, batch_size=32))
classifier = apply_classifier(inputs_np, targets_np, classifier)
```

```
Accuracy:  0.8680555555555556
```

In [132]:
```
classifier = MultiOutputClassifier(RandomForestClassifier())
classifier = apply_classifier(inputs_np, targets_np, classifier)
```

```
Accuracy:  0.8958333333333334

/home/anton/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:
245: FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [ ]:
```

```