In [170]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model       import LogisticRegression
import sklearn.discriminant_analysis as skl_da
import sklearn.neighbors as skl_nb
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn.model_selection    import train_test_split
from sklearn.preprocessing      import StandardScaler
from sklearn.metrics            import classification_report
from matplotlib                 import pyplot as plt
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.feature_selection import f_classif,mutual_info_classif, SelectKBest,f_regres
sion
from sklearn.metrics import log_loss
from sklearn import metrics
import sklearn.model_selection as skl_ms


np.random.seed(100)


#preparing data
music_data = pd.read_csv('training_data.csv')
music_test = pd.read_csv('songs_to_classify.csv')
```

Trying to study the data and the correlation between features using heatmap

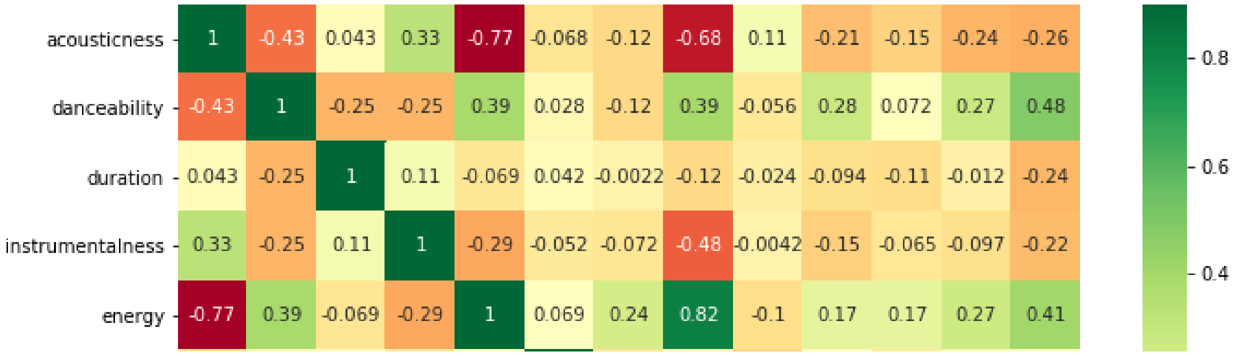
In [171]:

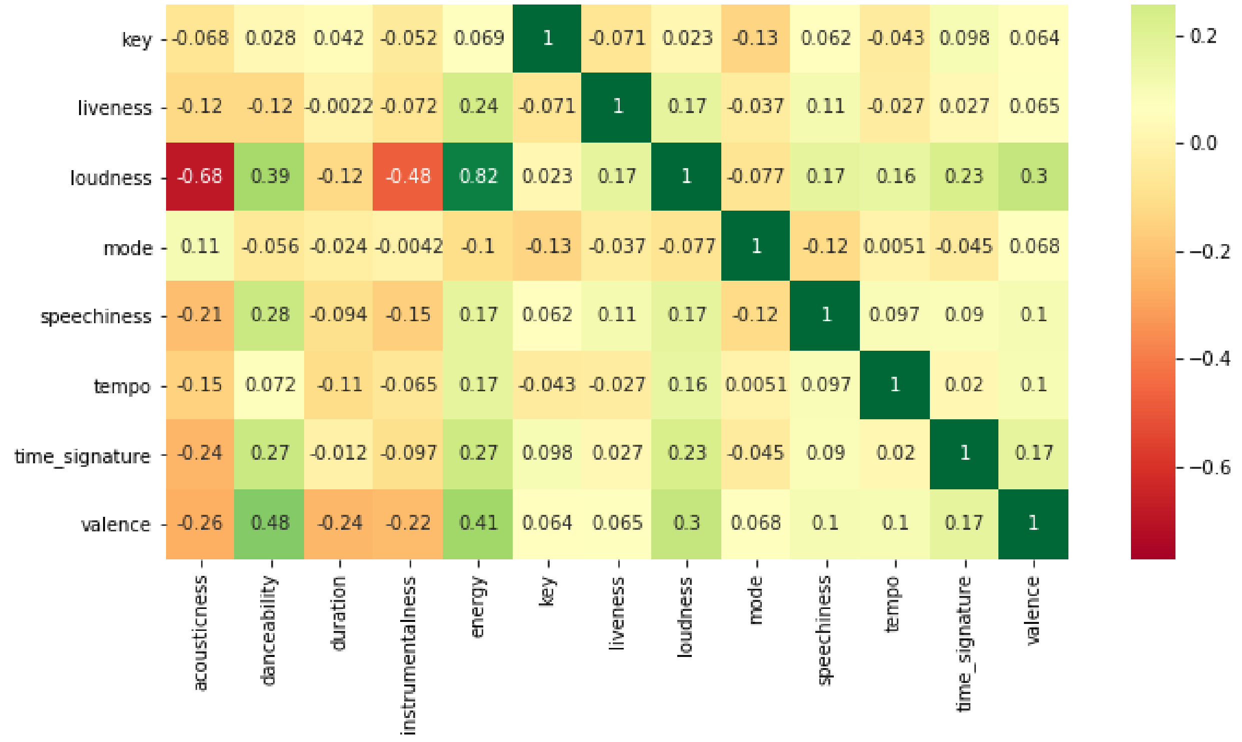
```
features = [
    "acousticness",
    "danceability",
    "duration",
    "instrumentalness",
    "energy",
    "key",
    "liveness",
    "loudness",
    "mode",
    "speechiness",
    "tempo",
    "time_signature",
    "valence" ]

m_data = music_data[features].corr()
plt.subplots(figsize=(12,9))
sns.heatmap(m_data,cmap="RdYlGn", annot=True,vmax=0.9, square=True)
```

Out[171]:

```
<matplotlib.axes._subplots.AxesSubplot object at 0x2a2750f10>
```





Trying to check each feature effectness on Like and dislike

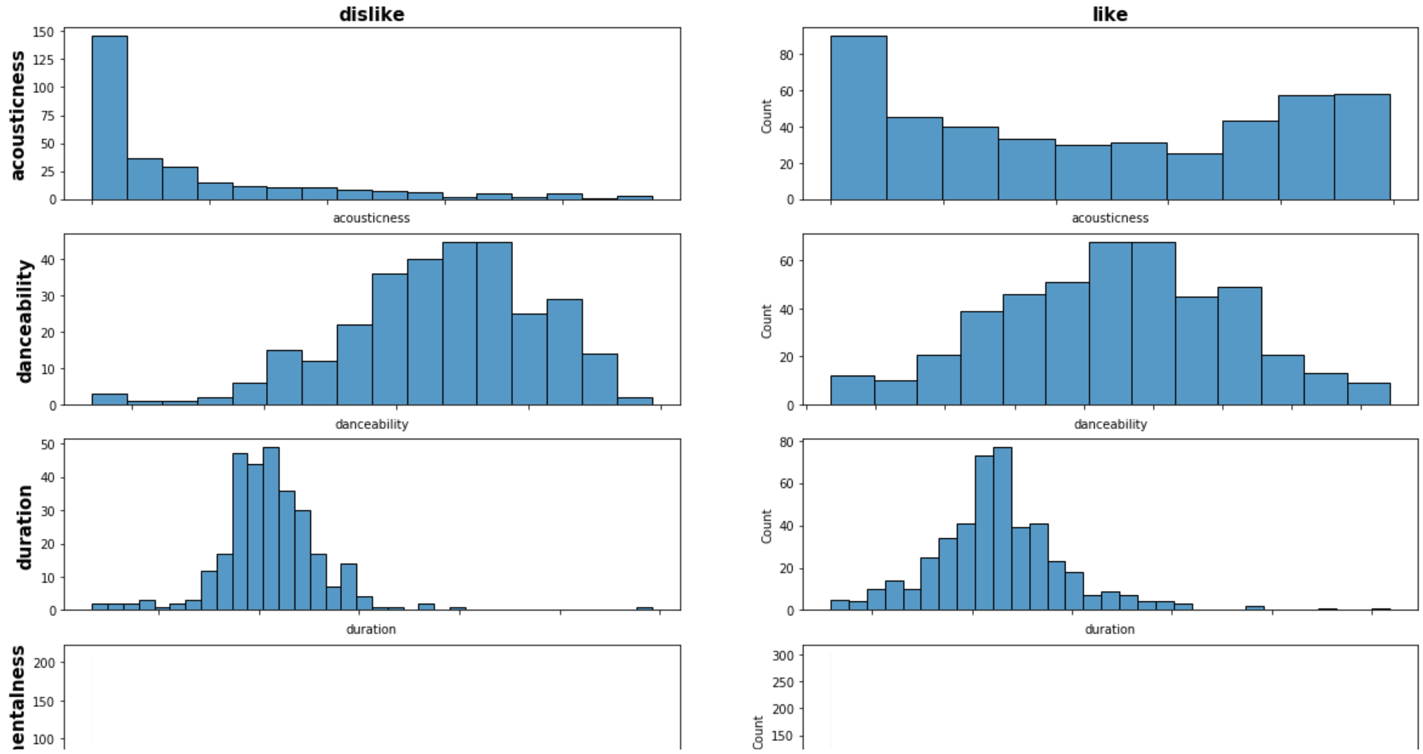
In [172]:

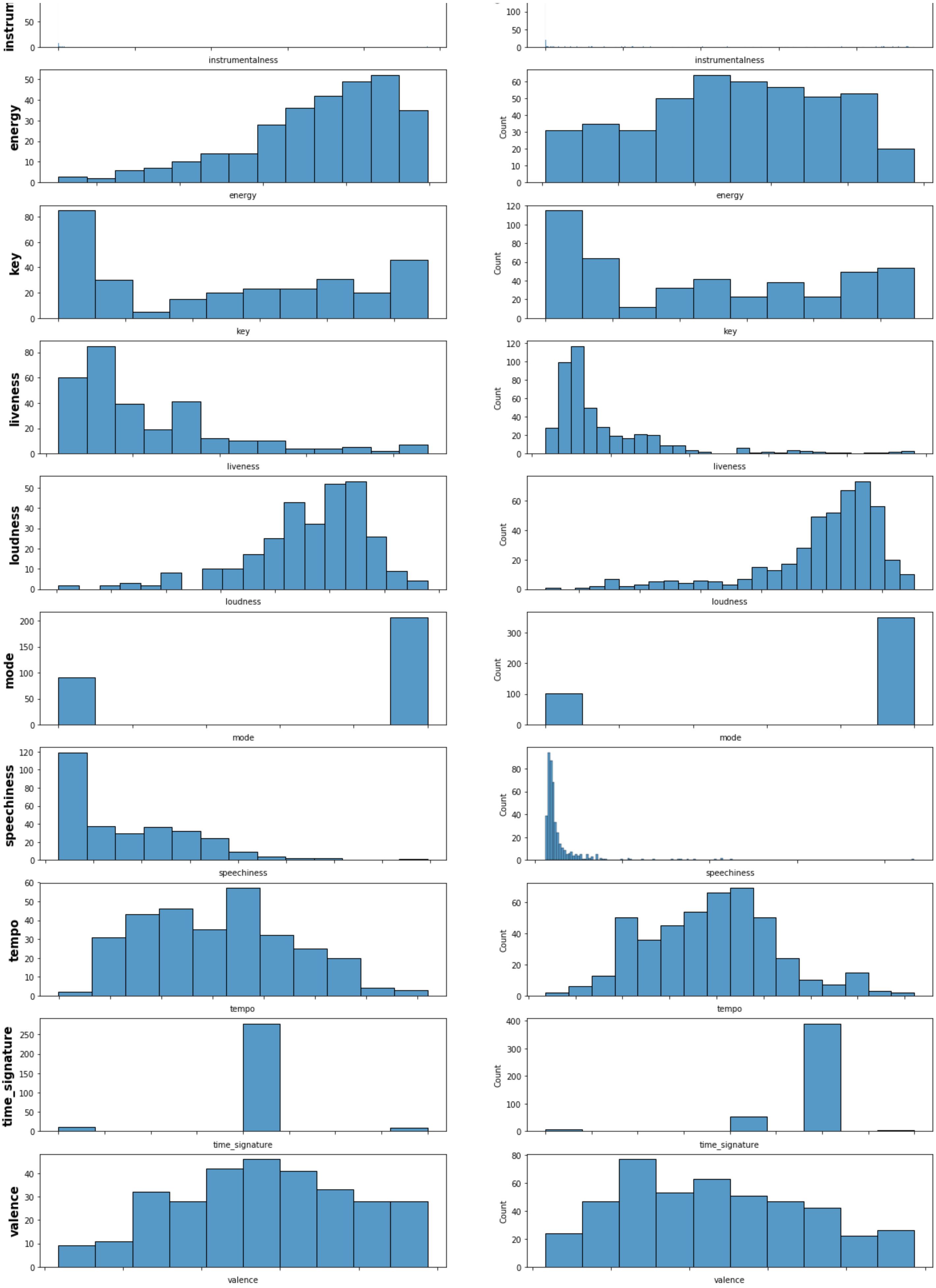
```
#separate like and dislike
dislike = music_data.loc[music_data['label'] == 0].copy()
like = music_data.loc[music_data['label'] == 1].copy()

fig, axs = plt.subplots(len(features), 2, figsize=(20, 40))

# Plotting histogram for eavh feature
axs[0,0].set_title('dislike', fontweight="bold", size=15)
axs[0,1].set_title('like', fontweight="bold", size=15)
for index, col in enumerate(features):
    axs[index,0].set_ylabel(col, fontweight="bold", fontsize=15)
    sns.histplot(dislike[col], ax=axs[index,0])
    sns.histplot(like[col], ax=axs[index,1])
    axs[index,0].set_xticklabels([])
    axs[index,1].set_xticklabels([])

plt.show()
```





We noticed that some features has changes in the data distribution in case of like and dislike and some dosn't have noticable change example of features that changed:

1. **acousticness**
2. **danceability**
3. **energy**

4. **liveness**
5. **oudness**
6. **speechiness**
7. **tempo**
8. **valence**
9. **duration**

WE will try to find a method for selecting the most important features for training the model

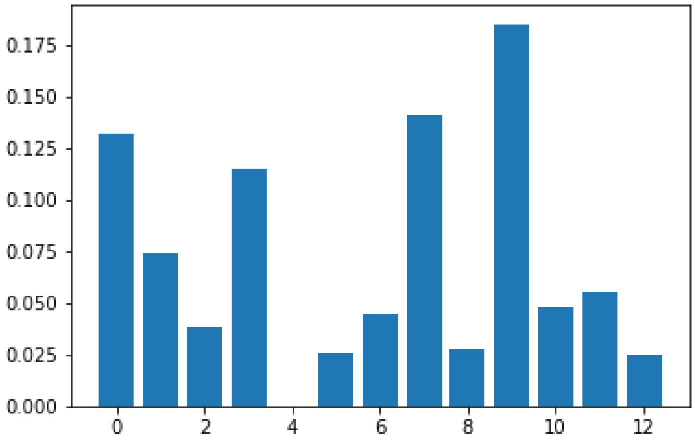
In [173]:

```
sel_f = SelectKBest(mutual_info_classif, k=9)
X_train_f = sel_f.fit(X_input, y_output)
print(sel_f.get_support())

for i in range(len(sel_f.scores_)):
 print('%s: %f' % (features[i], sel_f.scores_[i]))
# plot the scores
plt.bar([i for i in range(len(sel_f.scores_))], sel_f.scores_)
plt.show()

np.shape(X_train_f)
```

```
[ True  True  True  True False False  True  True False  True  True  True
 False]
acousticness: 0.132118
danceability: 0.073618
duration: 0.038445
instrumentalness: 0.114977
energy: 0.000000
key: 0.025767
liveness: 0.044364
loudness: 0.140876
mode: 0.027320
speechiness: 0.185026
tempo: 0.048318
time_signature: 0.054901
valence: 0.024419
```



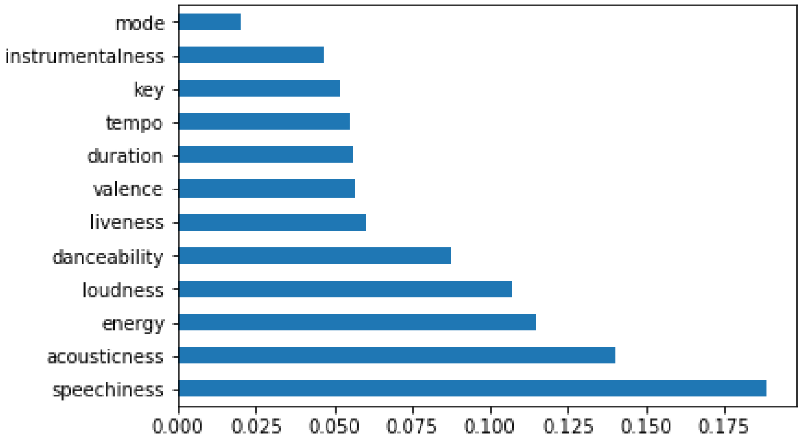
Out[173]:

```
()
```

In [174]:

```
from sklearn.ensemble import ExtraTreesClassifier
extr = ExtraTreesClassifier()
extr.fit(X_input, y_output)
print(extr.feature_importances_)
feat_importances = pd.Series(extr.feature_importances_, index=X_input.columns)
feat_importances.nlargest(12).plot(kind='barh')
plt.show()
```

```
[0.14028947 0.08747589 0.05621527 0.11488037 0.04651137 0.05191509
 0.06020034 0.10721684 0.02019557 0.18855259 0.05481158 0.01472643
 0.05700919]
```



Selected features which give highest accuracy

In [175]:

```
selected_features = ["acousticness",
    "danceability",
    "energy",
    "liveness",
    "loudness",
    "speechiness",
    "tempo",
    "valence"]
```

In [176]:

```
w = music_data[selected_features]
all_features = music_data[features]
m_labels = music_data['label'].copy()

scal=pd.DataFrame(ss.fit_transform(w), columns=[selected_features])
```

In [177]:

```
def trimWhiteSpace(arr):
    st=""
    for index in range(len(arr)):
        st+=str(arr[index])

    return st;
```

In [178]:

```
m_features_train, m_features_test, m_labels_train, m_labels_test = train_test_split(w, m
_labels, test_size=0.2)
baggingModel = BaggingClassifier()
baggingModel.fit(w, m_labels)
prediction = baggingModel.predict(music_test[selected_features])

print(skl_ms.cross_val_score(baggingModel, X_input , y_output , cv=10, scoring='accuracy
').mean())

print(trimWhiteSpace(prediction))
```

```
0.828
11100101011110011000101111111001110011111010110000000110100010001100001110010000101111110
01111000111101000101011001101101110000101101111110111111010100000000100101000010111111110
11101010101001111111000
```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
```