In [1]:

```
import pandas as pd
import numpy as np
from Develop.EMD2D import EMD2D
import cv2
from sklearn.preprocessing import minmax_scale
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

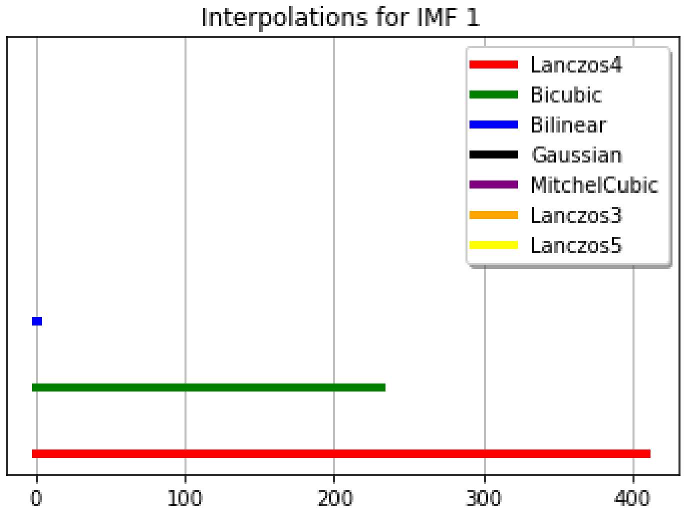
```
df = pd.read_csv('Interpolations.csv')
df = df.drop(columns=['Channels'])
df = df.apply(lambda x: x.astype('category') if x.dtype=='object' else
x)
to_work = df.copy()
interpolations = to_work['Interpolation Method'].unique().astype(str)
colors = ['r', 'g', 'b', 'black', 'purple', 'orange', 'yellow']
```

IMF Counter Plotting

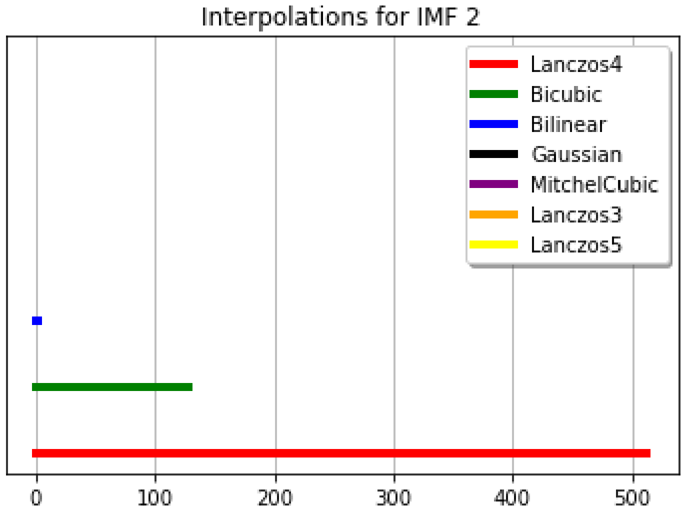
In [4]:

```
def imf_plot(imf: int):
    temp = to_work[to_work['IMF Spot'] == 'IMF ' + str(imf)]
    counts = np.array([])
    for i in range(len(interpolations)):
        x1 = temp[temp['Interpolation Method'] == interpolations[i]].co
unt()[0]
        x1 = np.linspace(0, x1, 2)
        y = np.repeat((i + 1) * 6, 2)
        counts = np.append(counts, plt.plot(x1, y, colors[i], linewidth
 = 4))
```

```
    plt.title('Interpolations for IMF ' + str(imf))
    plt.grid()
    plt.yticks([])
    plt.legend(counts, interpolations, fancybox=True, shadow=True, framealpha=1)
imf_plot(1)
```

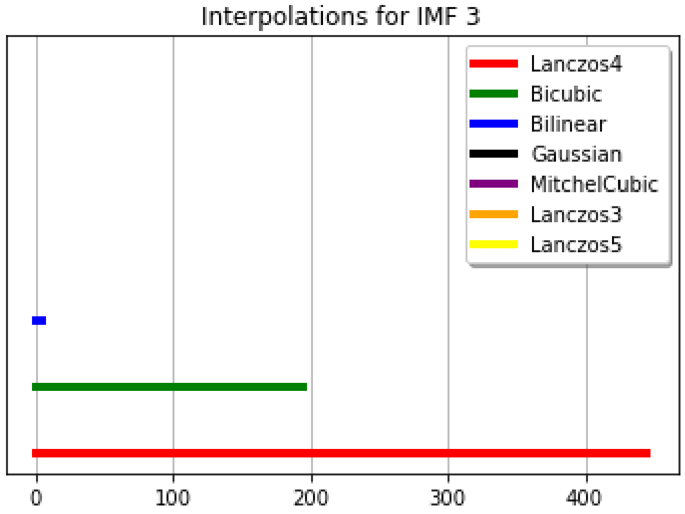


In [5]:
```
imf_plot(2)
```



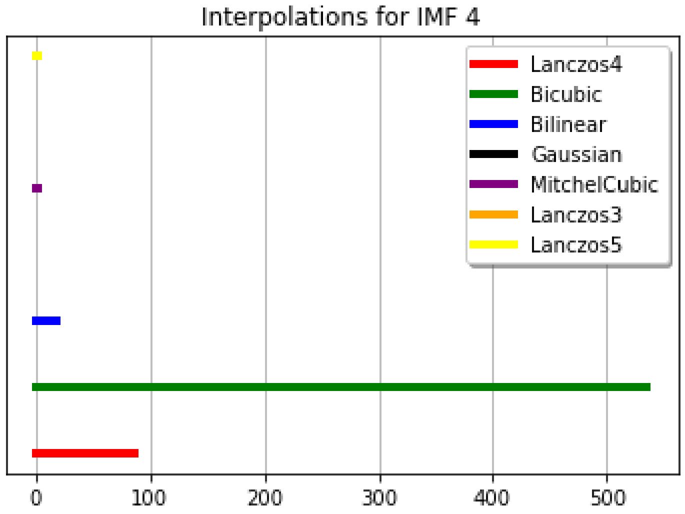
In [6]:

```
imf_plot(3)
```

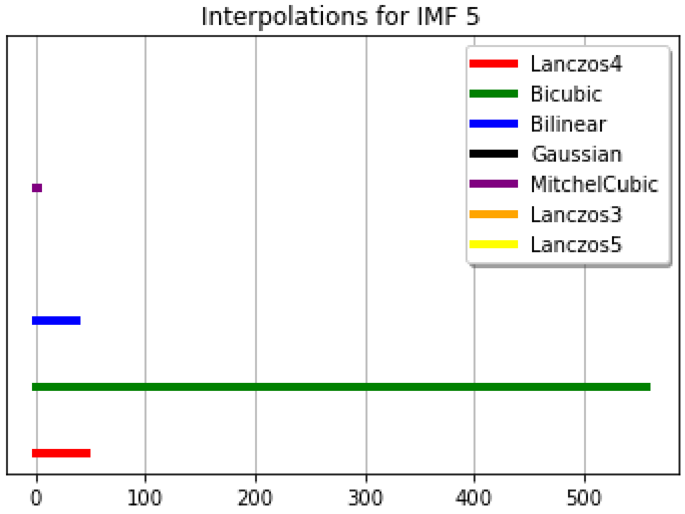


In [7]:

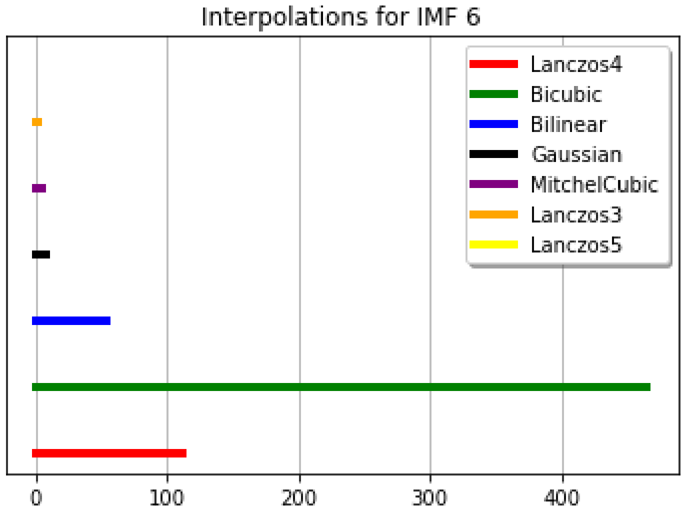
```
imf_plot(4)
```



In [8]:
```
imf_plot(5)
```

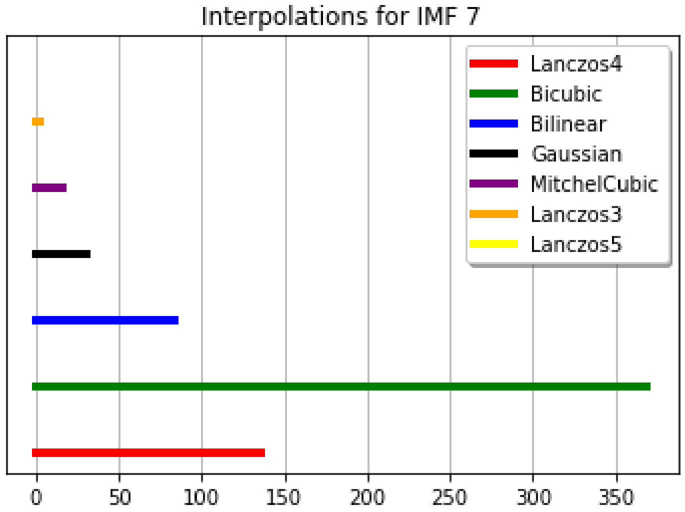


In [9]:
```
imf_plot(6)
```



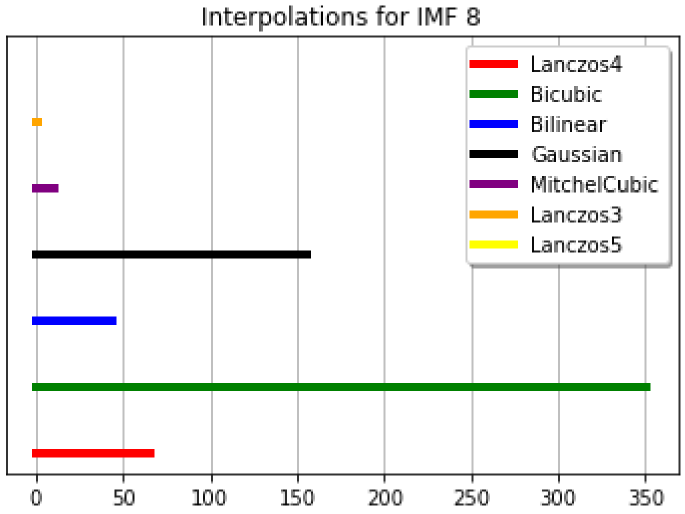
In [10]:

```
imf_plot(7)
```

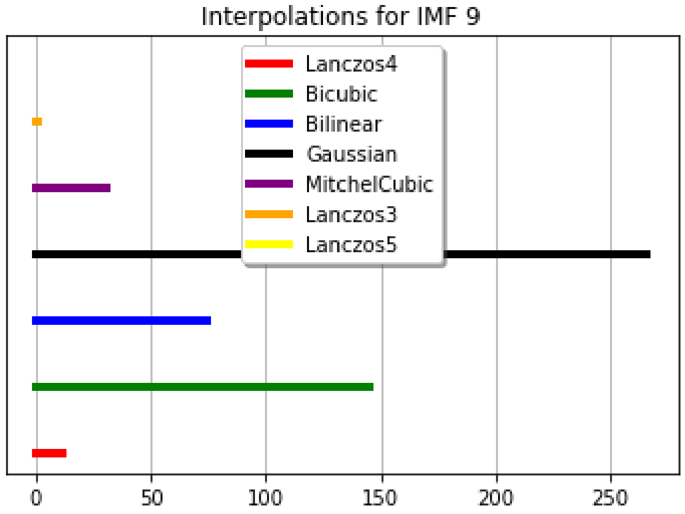


In [11]:

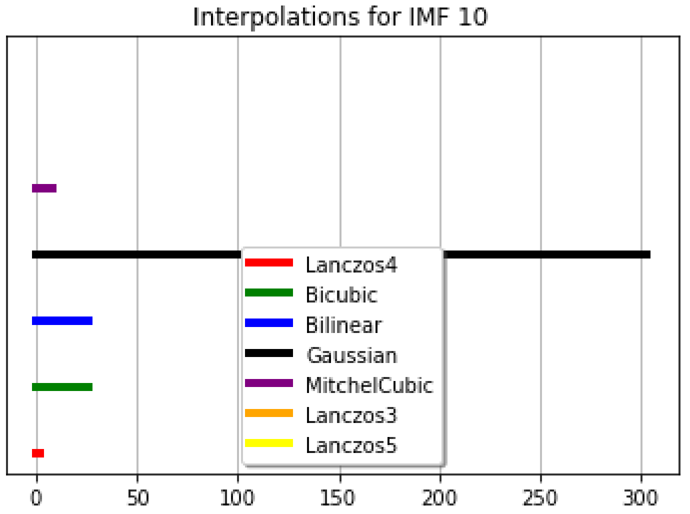
```
imf_plot(8)
```



```
In [12]: imf_plot(9)
```

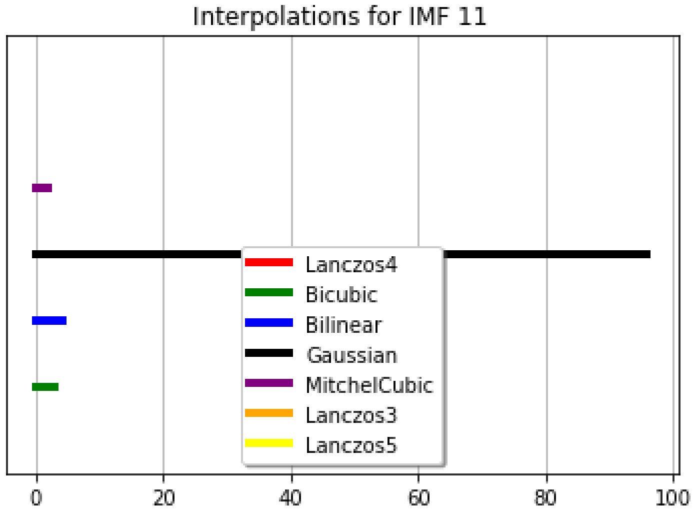


```
In [13]: imf_plot(10)
```



In [14]:

```
imf_plot(11)
```



In [15]:

```
to_work['IMF Spot'] = to_work['IMF Spot'].cat.codes
to_work['File Name'] = to_work['File Name'].cat.codes
to_work['Interpolation Method'] = to_work['Interpolation Method'].cat.codes
```

Defining Models + Train-Test Splitting

In [17]:
```
target = to_work['Interpolation Method']
to_work = to_work.drop(columns='Interpolation Method')

#to_work = minmax_scale(to_work)
#target = minmax_scale(target)
```

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(to_work, target)
```

In [19]:
```
random_forest = RandomForestClassifier()
random_forest.fit(x_train, y_train)
```

Out[19]:
```
RandomForestClassifier()
```

In [20]:
```
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
```

Out[20]:
```
KNeighborsClassifier()
```

In [21]:
```
desicion_tree = DecisionTreeClassifier()
desicion_tree.fit(x_train, y_train)
```

Out[21]:
```
DecisionTreeClassifier()
```

In [22]:
```
ada_boost = AdaBoostClassifier()
ada_boost.fit(x_train, y_train)
```

Out[22]:
```
AdaBoostClassifier()
```

In [23]:
```
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)
```

Out[23]:
```
LogisticRegression()
```

In [ ]:

In [ ]:

In [30]:

In [30]:

In [30]:

In [28]:

In [28]:

In [28]:

In [28]:

In [28]:

Test and Score Sun Oct 04 20, 12:44:55

Settings

Sampling type: No sampling, test on testing data
Target class: Average over classes

Scores

Model	AUC	CA	F1	Precision	Recall
kNN	0.876697266965202	0.6765868402697047	0.6500993728866614	0.651308820276103	0.6765868402697047
Tree	0.9934476529952697	0.9295512671471751	0.927764835805035	0.9286041605709752	0.9295512671471751
SVM	0.7300715140924828	0.45105789351313647	0.29867467180940027	0.6610092806230126	0.45105789351313647
Random Forest	0.9977776616161239	0.9600093001627529	0.9591428816007077	0.960616305858564	0.9600093001627529
Logistic Regression	0.8237104189535992	0.6921646128807254	0.6562405327230522	0.6527997386580694	0.6921646128807254
AdaBoost	1.0	1.0	1.0	1.0	1.0

We can tell that the AdaBoost model is over-fitting, the SVM model is not doing a very good job, but the other models, mostly Random Forest and Decision Tree, are doing much of a good job and are very precise.