In [143]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import seaborn as sns
import sklearn
from sklearn.preprocessing import scale
import sklearn.linear_model as skl_lm
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import statsmodels.formula.api as smf

train_df=pd.read_csv('train.csv')
y=train_df['label'].to_numpy()
var_train=train_df
del var_train['label']
X=var_train.to_numpy()
train_df
```

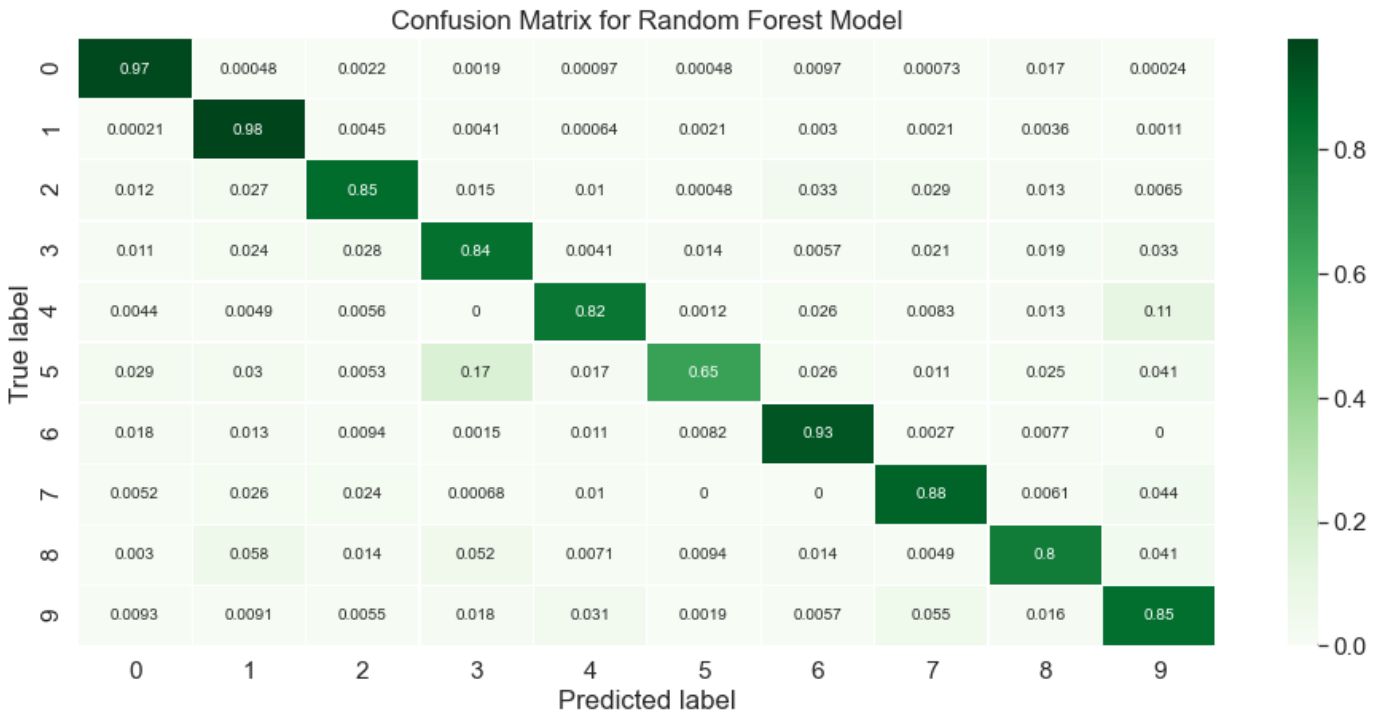
Out[143]:

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel7
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41995	0	0	0	0	0	0	0	0	0	0	...	0	0	
41996	0	0	0	0	0	0	0	0	0	0	...	0	0	
41997	0	0	0	0	0	0	0	0	0	0	...	0	0	
41998	0	0	0	0	0	0	0	0	0	0	...	0	0	
41999	0	0	0	0	0	0	0	0	0	0	...	0	0	

42000 rows × 784 columns

In [8]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn import tree
import datetime
from datetime import *

start=datetime.now()
forest = RandomForestClassifier(max_depth=5, random_state=0, n_estimators=150)
forest.fit(X, y)
accuracy_score(y, forest.predict(X))
confusion_matrix(y, forest.predict(X))
matrix = confusion_matrix(y, forest.predict(X))
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10}, cmap=plt.cm.Greens, linewidths=0.2)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
end=datetime.now()
print(end-start)
plt.show()
```

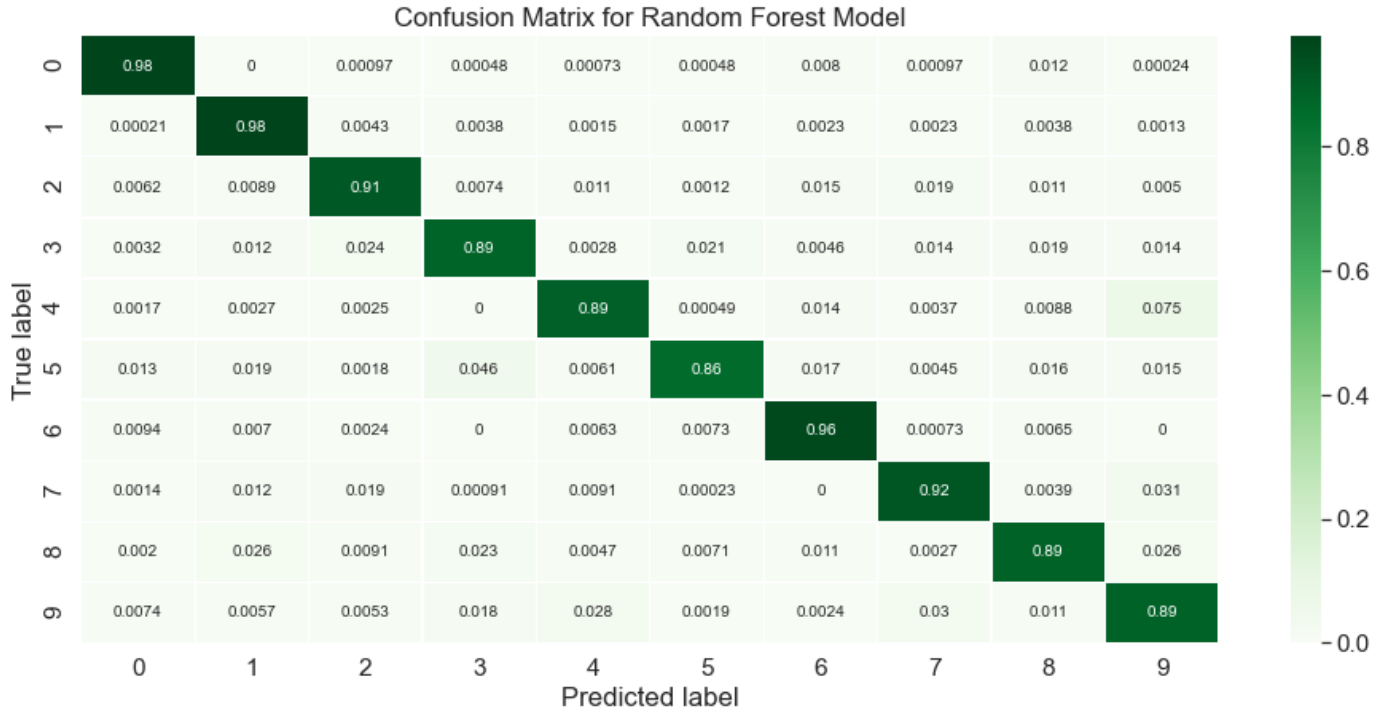
```
0:00:17.286893
```



In [ ]:
```
####^^^^^THIS SHOWS THAT THERE ARE ISSUES REDICTING THE NUMBER 5 AND PEOPLE ARE GETTING IT
###Trial and error optimization is below
```

In [9]:
```
start=datetime.now()
forest = RandomForestClassifier(max_depth=7, n_estimators=150, criterion='entropy', min_sar
forest.fit(X, y)
accuracy_score(y, forest.predict(X))
confusion_matrix(y, forest.predict(X))
matrix = confusion_matrix(y, forest.predict(X))
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10}, cmap=plt.cm.Greens, linewidths=0.2)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
end=datetime.now()
print(end-start)
plt.show()
```

```
0:00:36.756471
```



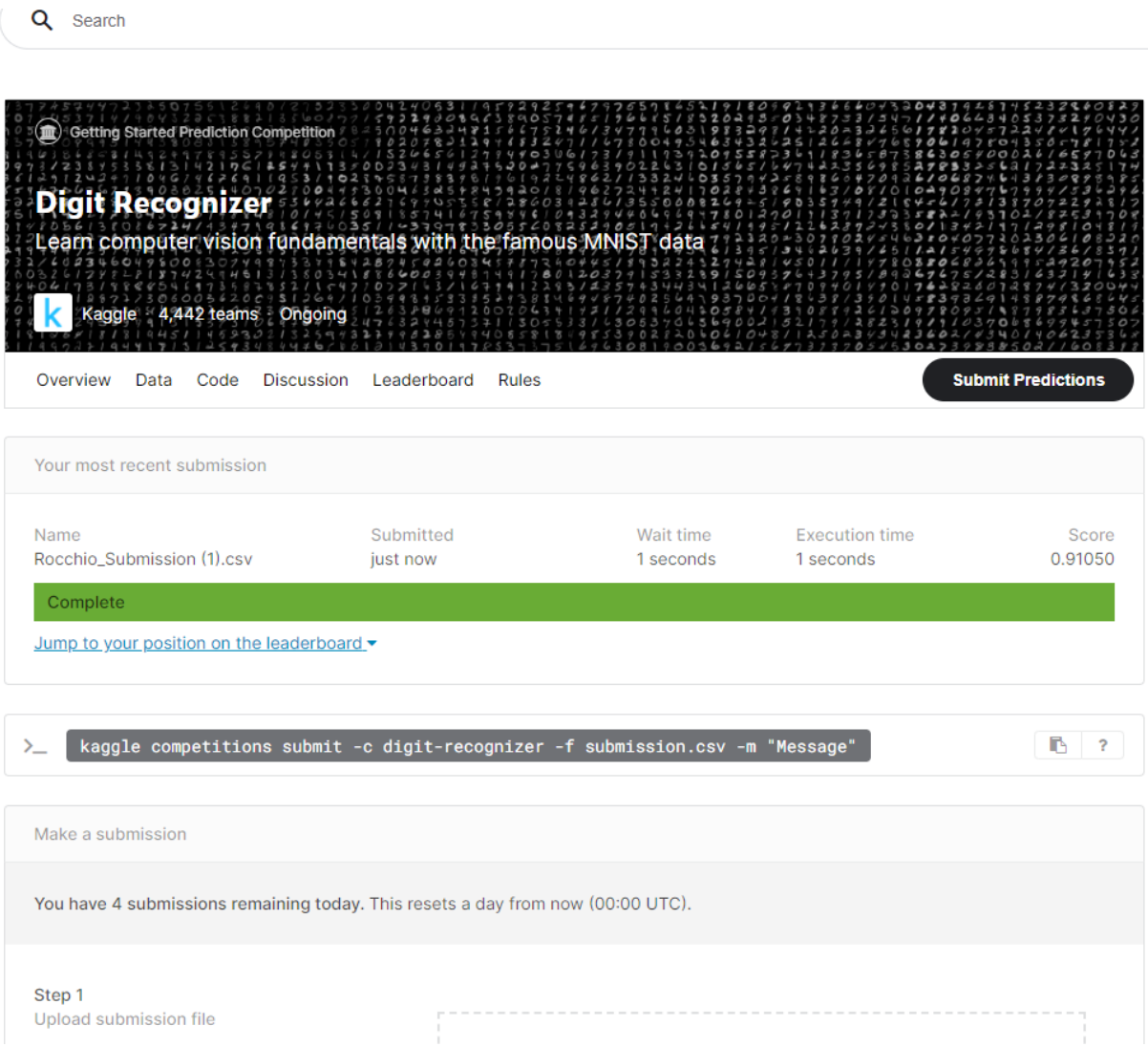
In [10]:
```
print(classification_report(y, forest.predict(X)))
#this looks promising
```

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      4132
           1       0.92      0.98      0.95      4684
           2       0.93      0.91      0.92      4177
           3       0.91      0.89      0.90      4351
           4       0.93      0.89      0.91      4072
           5       0.95      0.86      0.90      3795
           6       0.93      0.96      0.94      4137
           7       0.92      0.92      0.92      4401
           8       0.90      0.89      0.90      4063
           9       0.84      0.89      0.87      4188

    accuracy                           0.92     42000
   macro avg       0.92      0.92      0.92     42000
weighted avg       0.92      0.92      0.92     42000
```

In [12]:
```
test=pd.read_csv('test.csv')
x_test=test.to_numpy()
y_test=forest.predict(x_test)
submission=pd.read_csv('sample_submission.csv')
submission['Label']=y_test
submission.to_csv('Rocchio_Submission.csv', index=False)
```

In [35]:
```
from IPython.display import Image
Image(filename='kaggle_sub.png')

######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[35]:

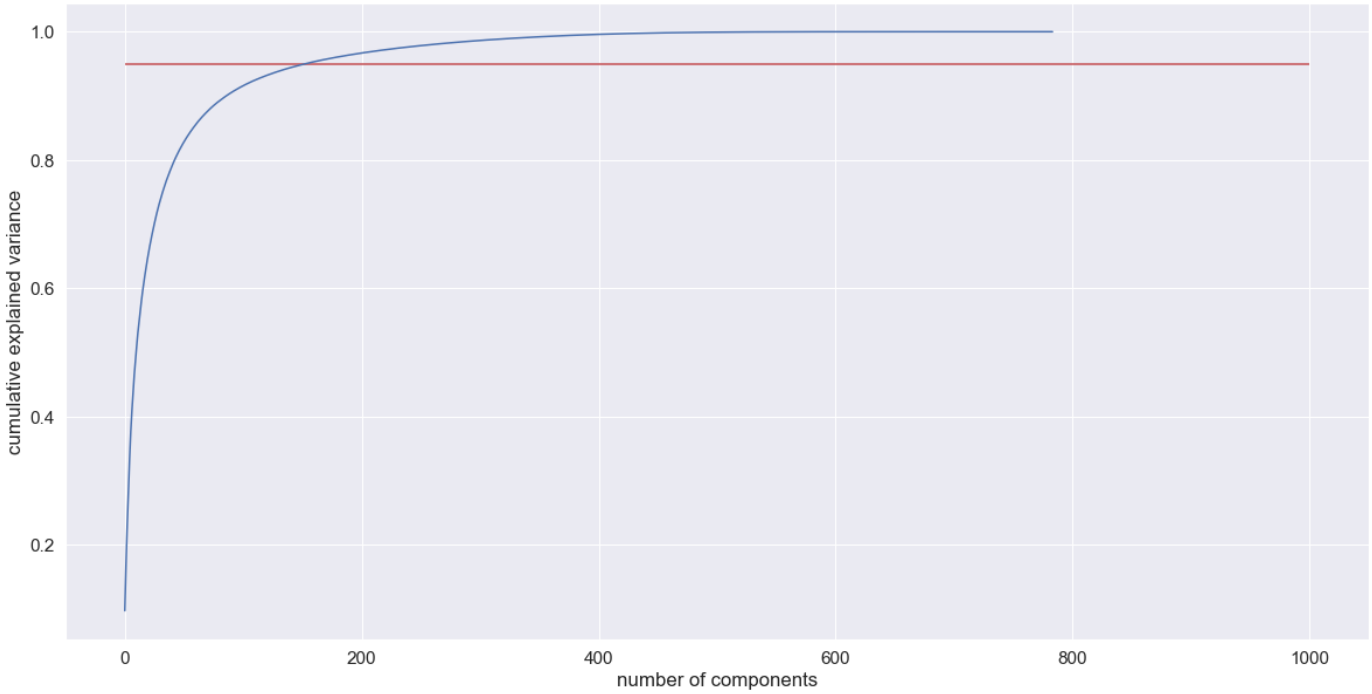


In [93]:
```
from sklearn.decomposition import PCA
start=datetime.now()

train_test_set=var_train.append(test)

X_pca = train_test_set.to_numpy()

pca = PCA(n_components=784)

pca.fit(X_pca)

print(pca.explained_variance_ratio_)

# print(pca.singular_values_)
X_pca
end=datetime.now()
print(end-start)
```

```
 4.71568617e-04 4.66713811e-04 4.64229952e-04 4.61992013e-04
 4.58284446e-04 4.49503116e-04 4.47172490e-04 4.40904677e-04
 4.37998438e-04 4.26374855e-04 4.20857295e-04 4.18174629e-04
 4.13001715e-04 4.08415056e-04 3.98256579e-04 3.93991596e-04
 3.91755679e-04 3.89121195e-04 3.83397936e-04 3.78486529e-04
 3.76442432e-04 3.72063436e-04 3.66321925e-04 3.64801469e-04

 3.62034223e-04 3.56831381e-04 3.53747734e-04 3.52293713e-04
 3.46618436e-04 3.44789075e-04 3.41853145e-04 3.38873721e-04
 3.34949967e-04 3.29111158e-04 3.27872188e-04 3.24171750e-04
 3.22790110e-04 3.20155820e-04 3.16615622e-04 3.15352396e-04
 3.09866885e-04 3.09377948e-04 3.05860527e-04 3.02546323e-04
 3.01222468e-04 3.00317395e-04 2.95516716e-04 2.95119362e-04
 2.92839952e-04 2.91751938e-04 2.86622584e-04 2.82929231e-04
 2.81562646e-04 2.77365996e-04 2.74347019e-04 2.72050408e-04
 2.68505641e-04 2.66949905e-04 2.64585298e-04 2.63632875e-04
 2.61567718e-04 2.58677209e-04 2.57444289e-04 2.56553991e-04
 2.54454214e-04 2.52751304e-04 2.51732760e-04 2.48492603e-04
 2.46879274e-04 2.45141268e-04 2.42557371e-04 2.41060642e-04
 2.40729101e-04 2.39157781e-04 2.38368958e-04 2.36465534e-04
```

In [94]:
```
import mplcursors
plt.figure(figsize=(20,10))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.hlines(.95, xmin=0, xmax=1000, color='r')
mplcursors.cursor(hover=True)
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
```

Out[94]:
```
Text(0, 0.5, 'cumulative explained variance')
```



In [95]:
```
###Based on above I would say that we need about 154 components to retain 95 percent of the
t=0
for i in np.cumsum(pca.explained_variance_ratio_):
    t=t+1
    if i>=.95:
        print(i, "n = ", t)
```

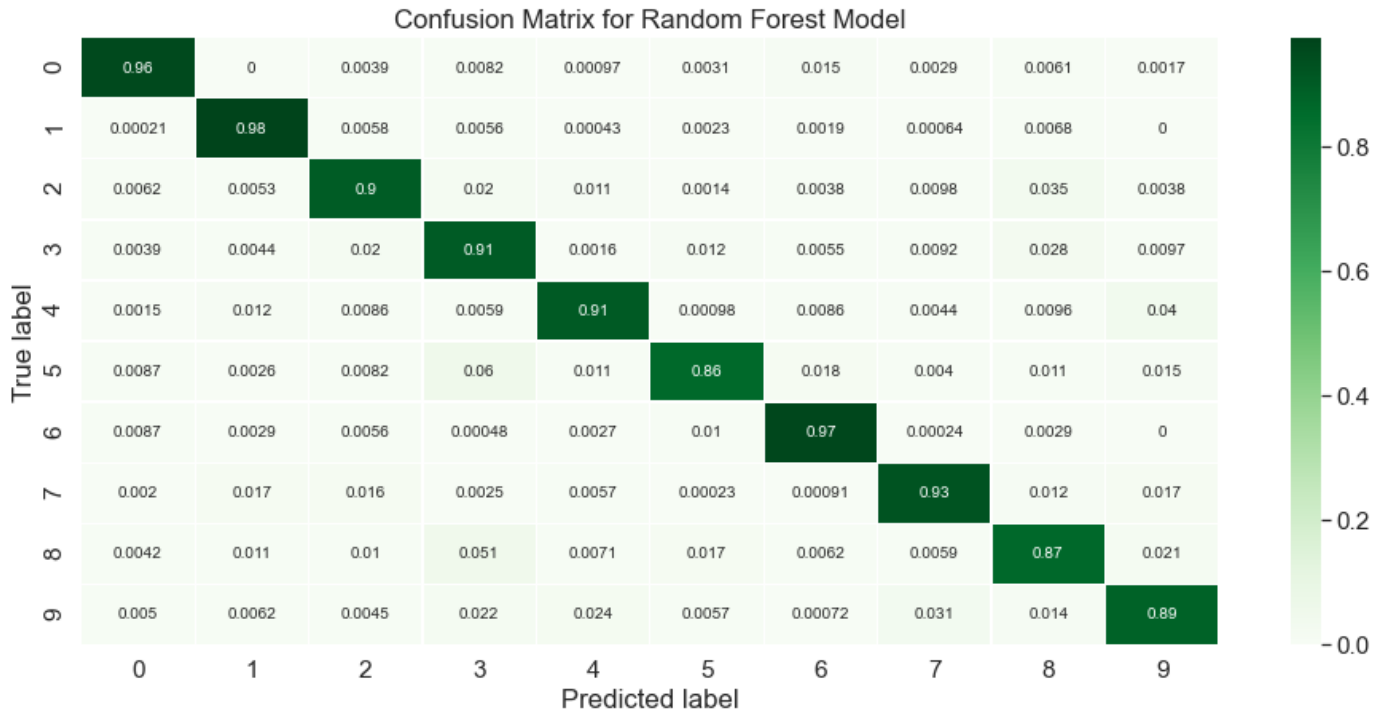
```
0.9503499702078612 n =  154
0.9507971426974613 n =  155
0.951238047374015 n =  156
0.9516760458119543 n =  157
0.9521024206666673 n =  158
0.9525232779613377 n =  159
0.952941452590341 n =  160
0.9533544543054279 n =  161
0.9537628693612 n =  162
0.9541611259400893 n =  163
0.9545551175356946 n =  164
0.9549468732145487 n =  165
0.955335994409155 n =  166
0.9557193923454379 n =  167
0.9560978788744631 n =  168
0.9564743213067269 n =  169
0.956846384742649 n =  170
0.9572127066671526 n =  171
0.9575775081360122 n =  172
```

In [92]:
```
pca.components_
```

Out[92]:
```
array([[-2.84106497e-19, -1.66533454e-16, -2.22044605e-16, ...,
        -0.00000000e+00, -0.00000000e+00, -0.00000000e+00],
       [-3.91144513e-19, -1.66533454e-16,  5.55111512e-17, ...,
        -0.00000000e+00, -0.00000000e+00, -0.00000000e+00],
       [ 1.24291198e-19, -1.11022302e-16, -0.00000000e+00, ...,
        -0.00000000e+00, -0.00000000e+00, -0.00000000e+00],
       ...,
       [ 0.00000000e+00, -1.38157076e-01, -1.70528790e-01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00,  5.32281332e-02,  2.34657228e-02, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00, -1.34264171e-02, -1.00319718e-01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00]])
```

In [132]:
```
##now I need to identify the 154 pixels containing the max variance prior to doing another
start=datetime.now()
train_df_rf2=pd.read_csv('train.csv')

label = train_df_rf2['label'].to_numpy()
del train_df_rf2['label']
X_fin_pca=train_df_rf2.to_numpy()
pca = PCA(n_components=154)

pca_train=pca.fit_transform(X_fin_pca)


forest_pca = RandomForestClassifier(max_depth=8, n_estimators=150, criterion='gini', min_sa
forest_pca.fit(pca_train, label)
accuracy_score(label, forest_pca.predict(pca_train))
confusion_matrix(label, forest_pca.predict(pca_train))
matrix = confusion_matrix(label, forest_pca.predict(pca_train))
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10}, cmap=plt.cm.Greens, linewidths=0.2)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
end=datetime.now()
print(end-start)
```

```
0:01:06.137198
```



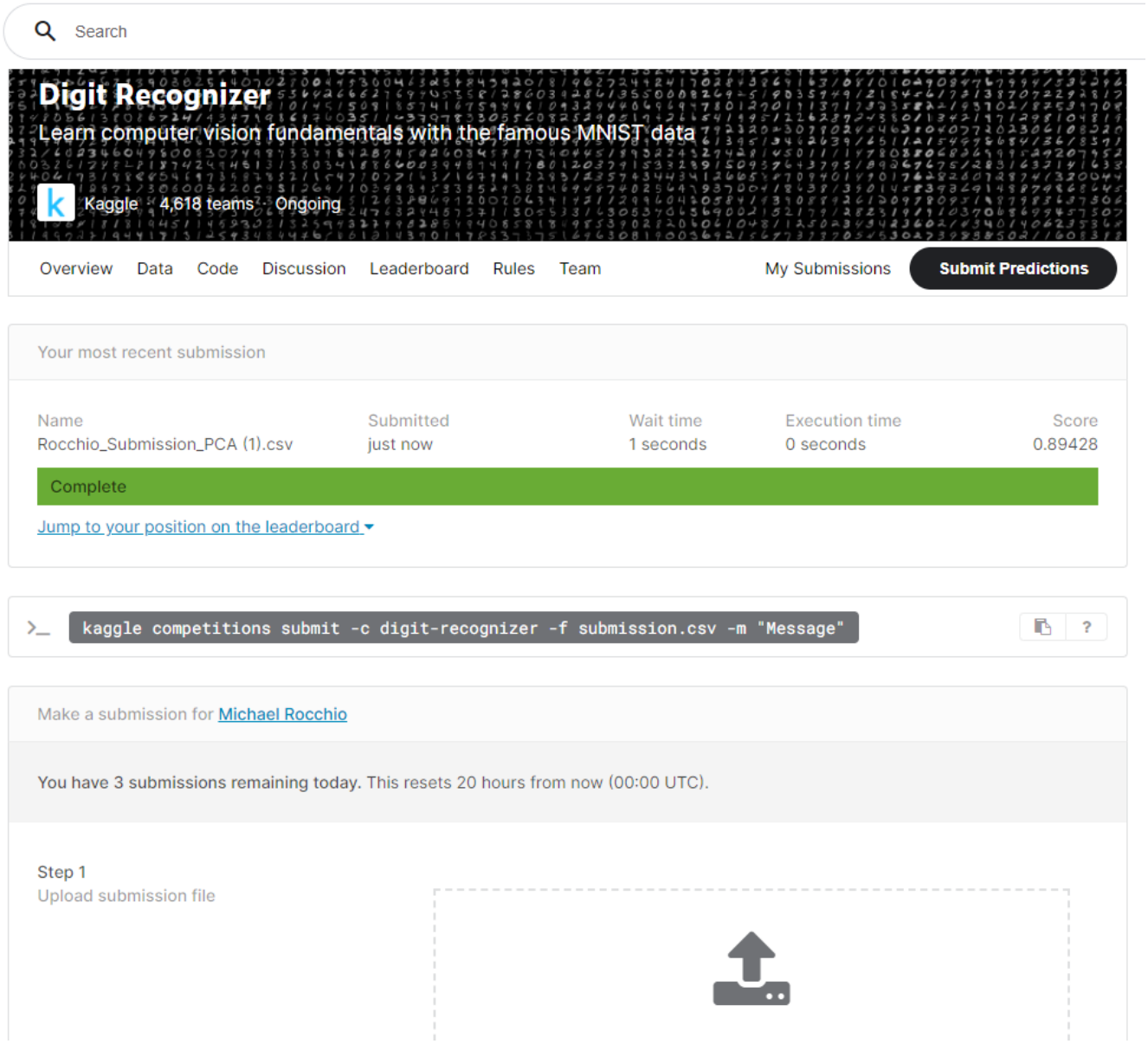
In [133]:
```
start=datetime.now()
test1=pd.read_csv('test.csv')
x_test1=test1.to_numpy()

pca_test=pca.transform(x_test1)


y_test=forest_pca.predict(pca_test)
submission=pd.read_csv('sample_submission.csv')
submission['Label']=y_test
submission.to_csv('Rocchio_Submission_PCA.csv', index=False)
end=datetime.now()
print(end-start)
y_test
```

```
0:00:02.851390
```

Out[133]:
```
array([2, 0, 9, ..., 3, 9, 2], dtype=int64)
```

In [134]:
```
from IPython.display import Image
Image(filename='kaggle_sub1.png')

######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[134]:



In [ ]:
```
#WE SHOULD NOT HAVE DONE DIMENSIONALITY REDUCTION ON A TEST SET WITH SO FEW NUMBERS
```

In [171]:
```
from sklearn.cluster import MiniBatchKMeans
from sklearn import metrics

def calculate_metrics(model,output):
 print('Number of clusters is {}'.format(model.n_clusters))
 print('Inertia : {}'.format(model.inertia_))
 print('Homogeneity :       {}'.format(metrics.homogeneity_score(output,model.labels_)))

def retrieve_info(cluster_labels,y_k):
    reference_labels = {}
    for i in range(len(np.unique(kmeans.labels_))):
        index = np.where(cluster_labels == i,1,0)
        num = np.bincount(y_k[index==1]).argmax()
        reference_labels[i] = num
    return reference_labels

start=datetime.now()

train_k=pd.read_csv('train.csv')

y_k = train_k['label'].to_numpy()
del train_k['label']
X_k=train_df_rf2.to_numpy()
X_k=X_k.astype(float)/255

Y=y_k
X=X_k

kmeans = MiniBatchKMeans(n_clusters = 10)

kmeans.fit(X_k)

calculate_metrics(kmeans,y_k)

reference_labels = retrieve_info(kmeans.labels_,y_k)

number_labels = np.random.rand(len(kmeans.labels_))
for i in range(len(kmeans.labels_)):

 number_labels[i] = reference_labels[kmeans.labels_[i]]

print('Accuracy score : {}'.format(accuracy_score(number_labels,y_k)))
print('\n')


# end=datetime.now()
# print(end-start)
```

```
Number of clusters is 10
Inertia : 1686468.2310341983
Homogeneity :       0.42477938280600447
Accuracy score : 0.5100238095238095
```

In [178]:

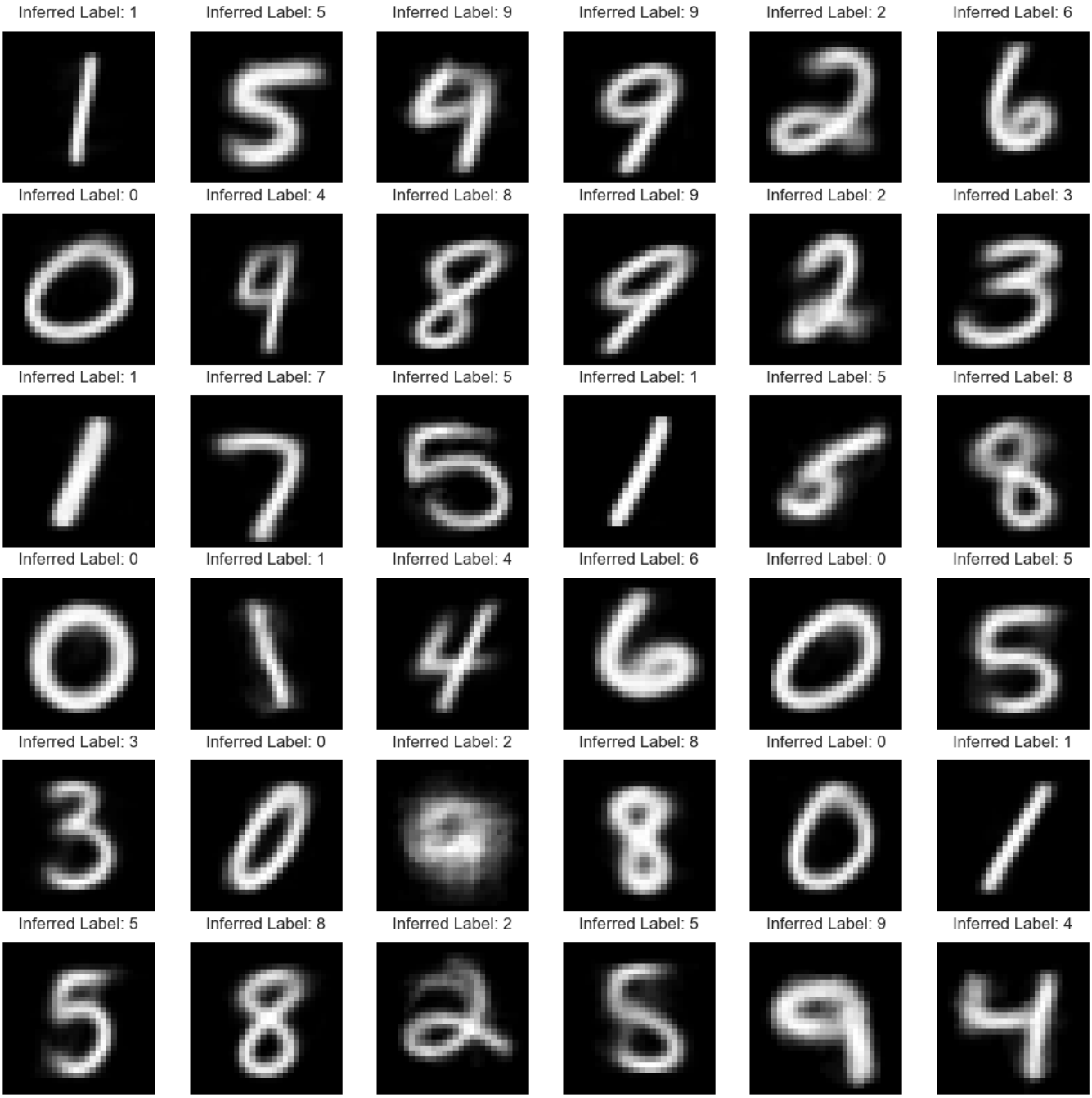
```
start=datetime.now()
clustering_number = [16,36,64,144,256,324, 484]

for i in clustering_number:
    kmeans = MiniBatchKMeans(n_clusters = i)

    kmeans.fit(X_k)

    calculate_metrics(kmeans,y_k)

    reference_labels = retrieve_info(kmeans.labels_,y_k)

    number_labels = np.random.rand(len(kmeans.labels_))
    for i in range(len(kmeans.labels_)):
        number_labels[i] = reference_labels[kmeans.labels_[i]]
    print('Accuracy score : {}'.format(accuracy_score(number_labels,y_k)))
    print('\n')

end=datetime.now()
print(end-start)
```

```
Number of clusters is 16
Inertia : 1545736.0914040292
Homogeneity :       0.5491175671584688
Accuracy score : 0.6386190476190476


Number of clusters is 36
Inertia : 1383208.3472508062
Homogeneity :       0.6786981666281809
Accuracy score : 0.7565714285714286


Number of clusters is 64
Inertia : 1269046.2253315833
Homogeneity :       0.7444599920156107
Accuracy score : 0.809404761904762


Number of clusters is 144
```

In [179]:
```
#print(labels)
 #print('Cluster: {}, label: {}'.format(i, np.argmax(counts)))

def infer_cluster_labels(kmeans, actual_labels):
    inferred_labels = {}

    for i in range(kmeans.n_clusters):
        labels = []
        index = np.where(kmeans.labels_ == i)
        labels.append(actual_labels[index])

        if len(labels[0]) == 1:
            counts = np.bincount(labels[0])
        else:
            counts = np.bincount(np.squeeze(labels))
        if np.argmax(counts) in inferred_labels:
            inferred_labels[np.argmax(counts)].append(i)
        else:
            inferred_labels[np.argmax(counts)] = [i]
#         print(labels)
#         print('Cluster: {}, label: {}'.format(i, np.argmax(counts)))
    return inferred_labels


def infer_data_labels(X_labels, cluster_labels):
    predicted_labels = np.zeros(len(X_labels)).astype(np.uint8)
    for i, cluster in enumerate(X_labels):
        for key, value in cluster_labels.items():
            if cluster in value:
                predicted_labels[i] = key
    return predicted_labels
```

In [182]:
```
start=datetime.now()
test=pd.read_csv('test.csv')
x_test=test.to_numpy()
x_test=x_test.astype(float)/255

centroids = kmeans.cluster_centers_
cluster_labels = infer_cluster_labels(kmeans, Y)
X_clusters = kmeans.predict(x_test)
predicted_labels = infer_data_labels(X_clusters, cluster_labels)
print(predicted_labels[:20])
print(Y[:20])
images = centroids.reshape(484, 28, 28)
images *= 255
images = images.astype(np.uint8)
cluster_labels = infer_cluster_labels(kmeans, Y)
fig, axs = plt.subplots(6, 6, figsize = (20, 20))
plt.gray()

for i, ax in enumerate(axs.flat):
    for key, value in cluster_labels.items():
        if i in value:
            ax.set_title('Inferred Label: {}'.format(key))
    ax.matshow(images[i])
    ax.axis('off')

fig.show()
end=datetime.now()
print(end-start)
```

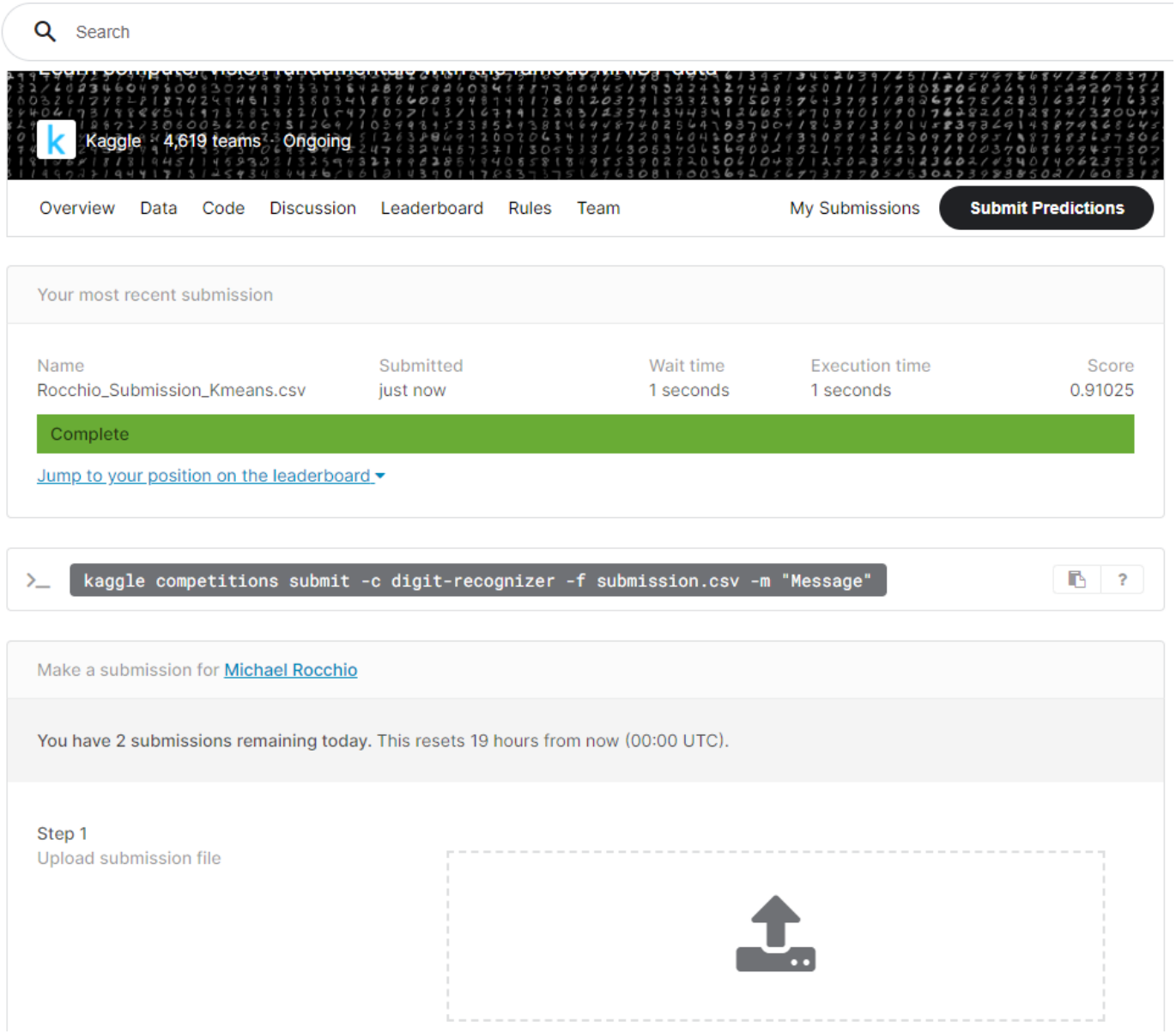
```
[2 0 9 4 3 7 0 3 0 3 5 7 9 0 4 5 3 1 9 0]
[1 0 1 4 0 0 7 3 5 3 8 9 1 3 3 1 2 0 7 5]

C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\ipykernel_launcher.py:25:
UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which
is a non-GUI backend, so cannot show the figure.
```



In [184]:
```
submission=pd.read_csv('sample_submission.csv')
submission['Label']=predicted_labels
submission.to_csv('Rocchio_Submission_Kmeans.csv', index=False)
```

In [185]:
```
Image(filename='kaggle_sub2.png')

######
######  MY USERNAME IN KAGGLE IS michaelrocchio ######
######
```

Out[185]:



In [193]:
```
# import sys
# !pip install LaTeX
# !pip install nbconvert
# !pip install Pandoc
# ! pip install nb_pdf_template
! pip install tex
! Jupyter nbconvert --to pdf Assignment_5.ipynb
```

```
Requirement already satisfied: tex in c:\users\rocchm1\appdata\roaming\python\python37\si
te-packages (1.8)

[NbConvertApp] Converting notebook Assignment_5.ipynb to pdf
Traceback (most recent call last):
  File "C:\Users\rocchm1\Anaconda3\Scripts\jupyter-nbconvert-script.py", line 10, in <m
odule>
    sys.exit(main())
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\jupyter_core\app
lication.py", line 270, in launch_instance
    return super(JupyterApp, cls).launch_instance(argv=argv, **kwargs)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\traitlets\config
\application.py", line 664, in launch_instance
    app.start()
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\nbconv
ertapp.py", line 350, in start
    self.convert_notebooks()
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\nbconv
ertapp.py", line 524, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\nbconv
ertapp.py", line 489, in convert_single_notebook
    output, resources = self.export_single_notebook(notebook_filename, resources, input
_buffer=input_buffer)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\nbconv
ertapp.py", line 418, in export_single_notebook
    output, resources = self.exporter.from_filename(notebook_filename, resources=resour
ces)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\exporter.py", line 181, in from_filename
    return self.from_file(f, resources=resources, **kw)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\exporter.py", line 199, in from_file
    return self.from_notebook_node(nbformat.read(file_stream, as_version=4), resources=
resources, **kw)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\pdf.py", line 169, in from_notebook_node
    nb, resources=resources, **kw
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\latex.py", line 77, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\templateexporter.py", line 384, in from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\templateexporter.py", line 148, in template
    self._template_cached = self._load_template()
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\nbconvert\export
ers\templateexporter.py", line 355, in _load_template
    return self.environment.get_template(template_file)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\jinja2\environme
nt.py", line 883, in get_template
    return self._load_template(name, self.make_globals(globals))
```

```
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\jinja2\environme
nt.py", line 857, in _load_template
    template = self.loader.load(self, name, globals)
  File "C:\Users\rocchm1\AppData\Roaming\Python\Python37\site-packages\jinja2\loaders.p
y", line 429, in load
    raise TemplateNotFound(name)
jinja2.exceptions.TemplateNotFound: index.tex.j2
```