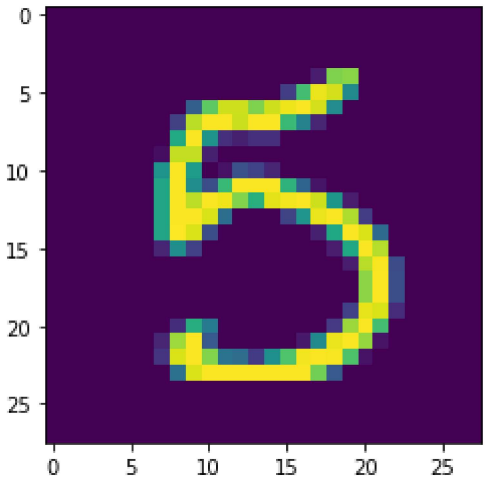
```
#Importing dependencies
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
%matplotlib inline


#using pandas to read database stored in the same folder
data =pd.read_csv('/content/sample_data/mnist_train_small.csv')


#extracting data from dataset and viewing them up close
a=data.iloc[3,1:].values


#viewing column heads
data.head()
```

	6	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	0.13	0.14	0.1
0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 785 columns

```
#reshaping the extracted data into a resonable size
a=a.reshape(28,28).astype('uint8')
plt.imshow(a)
```

```
<matplotlib.image.AxesImage at 0x7ff2e00e1860>
```



```
#preparing data
#seperating lables and data values
df_x=data.iloc[:,1:]
df_y=data.iloc[:,0]


#creating test and train sizez/batches
x_train,x_test,y_train,y_test=train_test_split(df_x,df_y,test_size=0.2,random_state=4)


#check data
y_train.head()
```

```
17914    2
16633    3
18252    3
17698    1
15617    1
Name: 6, dtype: int64
```

```
#call rf classifier
rf=RandomForestClassifier(n_estimators=100)


#fit the model
rf.fit(x_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
#prediction on test data
pred=rf.predict(x_test)


pred
```

```
array([0, 0, 0, ..., 1, 1, 8])
```

```
#check prediction accuracy
s=y_test.values
#calculate number of correctly predicted values
count=0
for i in range(len(pred)):
  if(pred[i]==s[i]):
   count=count+1


count
```

```
3823
```

```
#total values that the prediction code was run on
len(pred)
```

```
4000
```

```
#accuacy value
print(3823/4000*100,end='%')
```

```
95.575%
```