jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Code

Classification Using Random Forests (DF)

In [1]:
```
import os
from matplotlib import pyplot as plt
from sklearn.feature_extraction import DictVectorizer
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree
```

Acquire data in an accessible format

In [2]:
```
df = pd.read_csv(os.path.join(".", "Cleaned_Data", "default.csv"))
df.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	63	64	65	66
0	7.161286	7.835325	2.911583	0.984049	-1.499546	-2.094097	0.576000	-1.205671	1.849122	-0.425598	...	-0.174878	-1.089543	-0.668840	-0.914772	-0
1	0.225763	-0.094169	-0.603646	0.497745	0.874036	0.290280	-0.077659	-0.887385	0.432062	-0.093963	...	-0.157189	0.380951	1.088478	-0.123595	1
2	-0.692525	-0.517801	-0.788035	1.214351	-0.907214	0.880213	0.406899	-0.694895	-0.901869	-1.701574	...	2.718442	0.972919	2.081069	1.375763	1
3	-0.735562	-0.684055	2.058215	0.716328	-0.011393	0.805396	1.497982	0.114752	0.692847	0.052377	...	-1.020687	-0.751380	-0.385005	-0.012326	-0
4	0.570272	0.273157	-0.279214	0.083456	1.049331	-0.869295	-0.265858	-0.401676	-0.872639	1.147483	...	-0.190488	0.306974	0.119658	0.271838	1

5 rows × 73 columns

Prepare data for the machine learning model Using the 'Sub_Region' Column as Input

In [3]:
```
df.drop(columns=['Latitude', 'Longitude','Country', 'Region' ])
df = df.drop(columns=['Latitude', 'Longitude','Country', 'Region'])
```

In [4]:
```
import pandas as pd
Sub_Region_List = df['Sub_Region'].drop_duplicates()

print (Sub_Region_List)
```

```
0                South America
1               Western Africa
3               Eastern Africa
4              Northern Africa
8              Northern Europe
9                 Central Asia
11             Southern Europe
12               Southern Asia
15              Eastern Europe
29                Western Asia
34          South-eastern Asia
42                Eastern Asia
53                   Caribbean
76             Central America
97    Australia and New Zealand
Name: Sub_Region, dtype: object
```

jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Code

One-Hot Encoding

In [5]:
```
# Step 0: Reformat data
data = df.values
X = data[:, 0:67]
y = data[:, 68]
```

In [6]:
```
from sklearn.preprocessing import LabelEncoder

# Step 1: Label-encode data set
label_encoder = LabelEncoder()
label_encoder.fit(y)
encoded_y = label_encoder.transform(y)
```

In [7]:
```
from tensorflow.keras.utils import to_categorical

# Step 2: One-hot encoding
one_hot_y = to_categorical(encoded_y)
one_hot_y
```

Out[7]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [8]:
```
for label, original_class in zip(encoded_y, y):
    print('Original Class: ' + str(original_class))
    print('Encoded Label: ' + str(label))
    print('-' * 15)
```

```
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Northern Europe
Encoded Label: 8
---------------
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Eastern Europe
Encoded Label: 6
---------------
Original Class: Eastern Europe
Encoded Label: 6
```

Separate The Data into Features & Targets

In [9]:
```
target = one_hot_y
target_names = ["negative", "positive"]
```

In [10]:
```
data = df.drop("Sub_Region", axis=1)
feature_names = data.columns
data.head()
```

Out[10]:

	0	1	2	3	4	5	6	7	8	9	...	58	59	60	61
0	7.161286	7.835325	2.911583	0.984049	-1.499546	-2.094097	0.576000	-1.205671	1.849122	-0.425598	...	-0.944584	-0.043610	-1.504263	0.351267
1	0.225763	-0.094169	-0.603646	0.497745	0.874036	0.290280	-0.077659	-0.887385	0.432062	-0.093963	...	-0.082645	-0.947933	-0.495712	-0.465077
2	-0.692525	-0.517801	-0.788035	1.214351	-0.907214	0.880213	0.406899	-0.694895	-0.901869	-1.701574	...	-0.797954	-0.556109	-0.637167	0.147260
3	-0.735562	-0.684055	2.058215	0.716328	-0.011393	0.805396	1.497982	0.114752	0.692847	0.052377	...	-0.805626	0.166616	-0.178325	-0.065059
4	0.570272	0.273157	-0.279214	0.083456	1.049331	-0.869295	-0.265858	-0.401676	-0.872639	1.147483	...	-0.180181	-0.500785	-0.919463	-0.667912

jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help

Convert DataTypes For The Training & Testing Data Sets

In [11]:
```
df.dtypes
```

Out[11]:
```
0            float64
1            float64
2            float64
3            float64
4            float64
              ...
64           float64
65           float64
66           float64
67           float64
Sub_Region    object
Length: 69, dtype: object
```

In [12]:
```
# Convert 'Sub_Region' Column to Float

df["Sub_Region"] = pd.to_numeric(df.Sub_Region, errors='coerce' )
df.dtypes
```

Out[12]:
```
0            float64
1            float64
2            float64
3            float64
4            float64
              ...
64           float64
65           float64
66           float64
67           float64
Sub_Region   float64
Length: 69, dtype: object
```

Train and split Model with random forest regression model from skicit-learn

In [13]:
```
# Import  model
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 1234)
# Split Data Into Testing and Training Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, target, random_state=1234)
```

Train & Split Model with Random Forest Classification Model

In [14]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=200)
rf = rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

Out[14]:
```
0.07547169811320754
```

Shape of all the Data

In [15]:
```
print('X_train Shape:', X_train.shape)
print('y_train Shape:', y_train.shape)
print('X_test Shape:', X_test.shape)
print('y_test Shape:', y_test.shape)
```

```
X_train Shape: (794, 68)
y_train Shape: (794, 15)
X_test Shape: (265, 68)
y_test Shape: (265, 15)
```

jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help

Metrics and Scoring For Classification Model

In [16]:
```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[16]:
```
0.3018867924528302
```

In [17]:
```
# Train the model on training data
rf = rf.fit(X_train, y_train);
rf.score(X_test, y_test)
```

Out[17]:
```
0.09433962264150944
```

Make Predictions & Calculate Errors

In [18]:
```
# Use numpy to convert to arrays
import numpy as np
# Use the forest's predict method on the test data
predictions = rf.predict(X_test)
predictions
```

Out[18]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [19]:
```
# Use the forest's predict method on the test data
predictions = rf.predict(X_test)
# Calculate the absolute errors
errors = abs(predictions - y_test,  )
errors
```

Out[19]:
```
array([[0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [20]:
```
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 0.06 degrees.
```

In [21]:
```
# Calculate mean absolute percentage error (MAPE)
# mape = 100 * (errors / y_test, )
# Calculate and display accuracy
# accuracy = 100 - np.mean(mape, )
# print('Accuracy:', round(accuracy, 2), '%.')
```

In [22]:
```
sorted(zip(rf.feature_importances_, target_names), reverse=True)
```

Out[22]:
```
[(0.01675877242647694, 'positive'), (0.013195868470672337, 'negative')]
```

Visualizing The Decision Tree in Regression Task

In [23]:
```
# Fit the regressor, set max_depth = 3
regr = DecisionTreeRegressor(max_depth=3, random_state=1234)
model = regr.fit(X, one_hot_y)
```

In [24]:
```
text_representation = tree.export_text(regr)
print(text_representation)
```

```
|--- feature_52 <= 1.22
|   |--- feature_66 <= 0.89
```

jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help

Visualizing The Decision Tree in Regression Task

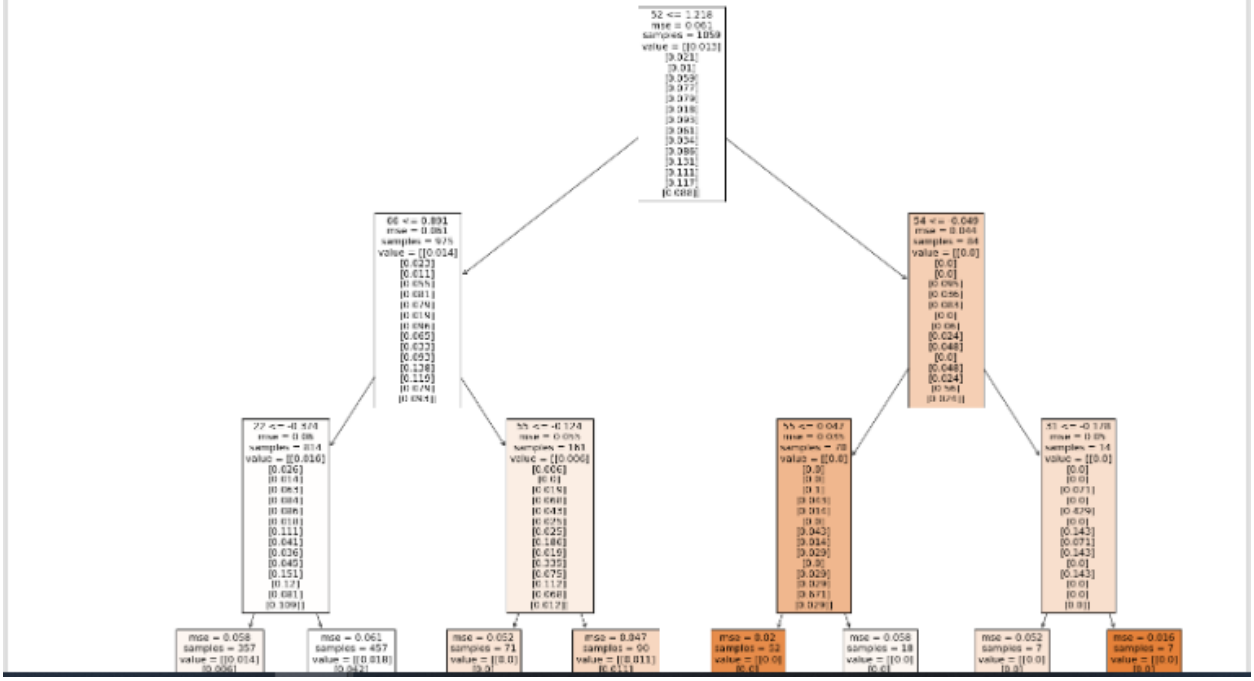
In [23]:
```
# Fit the regressor, set max_depth = 3
regr = DecisionTreeRegressor(max_depth=3, random_state=1234)
model = regr.fit(X, one_hot_y)
```

In [24]:
```
text_representation = tree.export_text(regr)
print(text_representation)
```

```
|--- feature_52 <= 1.22
|   |--- feature_66 <= 0.89
|   |   |--- feature_22 <= -0.37
|   |   |   |--- value: [0.01, 0.01, 0.00, 0.05, 0.03, 0.13, 0.01, 0.10, 0.04, 0.02, 0.04, 0.24, 0.12, 0.03, 0.18]
|   |   |--- feature_22 >  -0.37
|   |   |   |--- value: [0.02, 0.04, 0.02, 0.07, 0.12, 0.05, 0.03, 0.12, 0.04, 0.05, 0.05, 0.09, 0.12, 0.12, 0.06]
|   |--- feature_66 >  0.89
|   |   |--- feature_55 <= -0.12
|   |   |   |--- value: [0.00, 0.00, 0.00, 0.04, 0.10, 0.07, 0.00, 0.01, 0.42, 0.00, 0.11, 0.10, 0.06, 0.07, 0.01]
|   |   |--- feature_55 >  -0.12
|   |   |   |--- value: [0.01, 0.01, 0.00, 0.00, 0.04, 0.02, 0.04, 0.03, 0.00, 0.03, 0.51, 0.06, 0.16, 0.07, 0.01]
|--- feature_52 >  1.22
|   |--- feature_54 <= -0.05
|   |   |--- feature_55 <= 0.05
|   |   |   |--- value: [0.00, 0.00, 0.00, 0.12, 0.00, 0.00, 0.00, 0.02, 0.00, 0.00, 0.00, 0.02, 0.00, 0.83, 0.02]
|   |   |--- feature_55 >  0.05
|   |   |   |--- value: [0.00, 0.00, 0.00, 0.06, 0.17, 0.06, 0.00, 0.11, 0.06, 0.11, 0.00, 0.06, 0.11, 0.22, 0.06]
|   |--- feature_54 >  -0.05
|   |   |--- feature_31 <= -0.18
|   |   |   |--- value: [0.00, 0.00, 0.00, 0.14, 0.00, 0.00, 0.00, 0.29, 0.14, 0.29, 0.00, 0.14, 0.00, 0.00, 0.00]
|   |   |--- feature_31 >  -0.18
|   |   |   |--- value: [0.00, 0.00, 0.00, 0.00, 0.00, 0.86, 0.00, 0.00, 0.00, 0.00, 0.00, 0.14, 0.00, 0.00, 0.00]
```

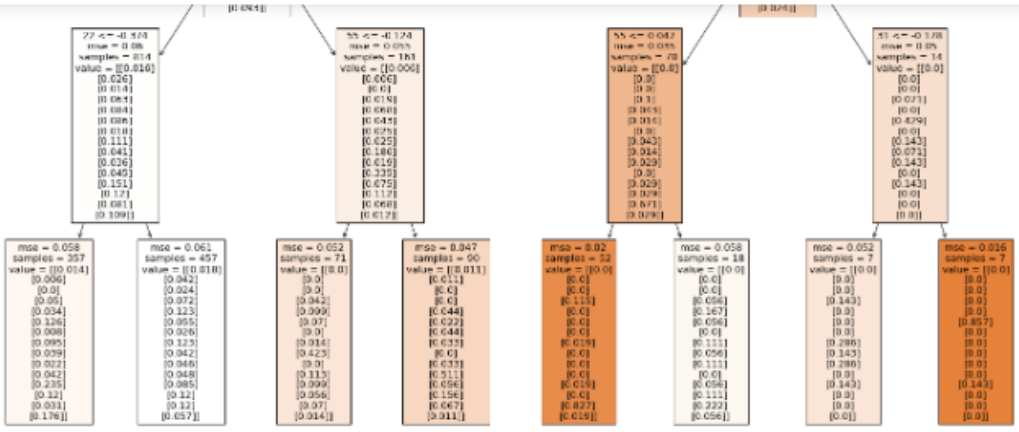
In [25]:
```
# Note that color of the leaf coresponds to the predicted value.
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(regr, feature_names=data.columns, filled=True)

fig.savefig("static/images/random_trees_DF")
```



jupyter Random_Trees_DF Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help



Random Trees Data Structure and Visualization

In [26]:
```
# Total Count of Nodes

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
n_nodes = rf.estimators_[0].tree_.node_count
n_nodes
```

Out[26]: 457

In [27]:
```
# Visualize (5) Random Trees
# Note ALSO that color of the leaf coresponds to the predicted value.
fn=data.columns
cn=one_hot_y

fig, axes = plt.subplots(nrows = 1,ncols = 5,figsize = (12,2), dpi=720)
for index in range(0, 5):
    tree.plot_tree(rf.estimators_[index],
                   feature_names = fn,
                   class_names=cn,
                   filled = True,
                   ax = axes[index]);

axes[index].set_title('Estimator: ' + str(index), fontsize = 11)

fig.savefig("static/images/rt_5trees-DF.png")
```

In [ ]: