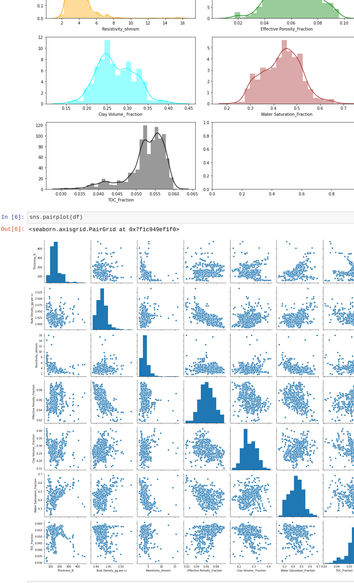
In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
df=pd.read_excel('DataSet.xlsx')
df.describe().transpose()
```

Out[4]:

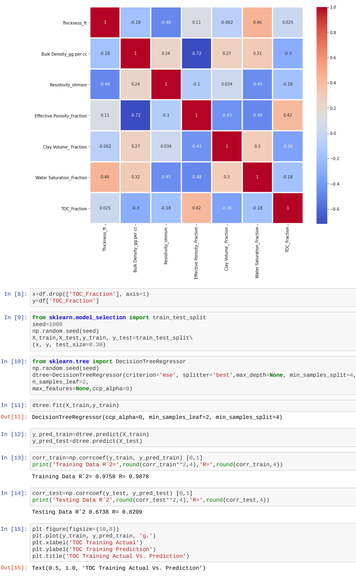
	count	mean	std	min	25%	50%	75%	max
Thickness_ft	987.0	150.448933	52.452284	50.218753	123.462354	141.662622	166.707110	475.992627
Bulk Density_gg per cc	987.0	2.423001	0.019059	2.385117	2.409449	2.422639	2.433418	2.540608
Resistivity_ohmsm	987.0	3.853432	1.342193	1.680451	3.120852	3.692354	4.319585	15.970635
Effective Porosity_Fraction	987.0	0.061492	0.014805	0.017432	0.051250	0.060158	0.072289	0.096054
Clay Volume_ Fraction	987.0	0.271257	0.045289	0.153318	0.238607	0.264785	0.303776	0.413083
Water Saturation_Fraction	987.0	0.435876	0.060023	0.230041	0.372234	0.442414	0.490972	0.683364
TOC_Fraction	987.0	0.052630	0.005062	0.030830	0.051026	0.053662	0.056100	0.060907

In [5]:
```
f, axes=plt.subplots(4, 2, figsize=(12, 12))
sns.distplot(df['Thickness_ft'], color="red", ax=axes[0, 0])
sns.distplot(df['Bulk Density_gg per cc'], color="blue", ax=axes[0, 1])
sns.distplot(df['Resistivity_ohmsm'], color="orange",ax=axes[1, 0])
sns.distplot(df['Effective Porosity_Fraction'], color="green",ax=axes[1, 1])
sns.distplot(df['Clay Volume_ Fraction'], color="cyan",ax=axes[2, 0])
sns.distplot(df['Water Saturation_Fraction'], color="brown",ax=axes[2, 1])
sns.distplot(df['TOC_Fraction'], color="black", ax=axes[3, 0])
plt.tight_layout()
```



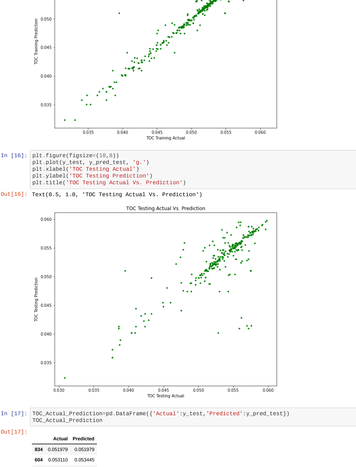
In [6]:
```
sns.pairplot(df)
```

Out[6]:
<seaborn.axisgrid.PairGrid at 0x7f1c949ef1f0>



In [7]:
```
plt.figure(figsize=(12,10))
sns.heatmap(df.corr(), annot=True, linecolor='white',
linewidths=2, cmap='coolwarm')
```

Out[7]:
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c9520d9a0>



In [8]:
```
x=df.drop(['TOC_Fraction'], axis=1)
y=df['TOC_Fraction']
```

In [9]:
```
from sklearn.model_selection import train_test_split
seed=1000
np.random.seed(seed)
X_train,X_test,y_train, y_test=train_test_split\
(x, y, test_size=0.30)
```

In [10]:
```
from sklearn.tree import DecisionTreeRegressor
np.random.seed(seed)
dtree=DecisionTreeRegressor(criterion='mse', splitter='best',max_depth=None, min_samples_split=4, mi
n_samples_leaf=2,
max_features=None,ccp_alpha=0)
```

In [11]:
```
dtree.fit(X_train,y_train)
```

Out[11]:
```
DecisionTreeRegressor(ccp_alpha=0, min_samples_leaf=2, min_samples_split=4)
```

In [12]:
```
y_pred_train=dtree.predict(X_train)
y_pred_test=dtree.predict(X_test)
```

In [13]:
```
corr_train=np.corrcoef(y_train, y_pred_train) [0,1]
print('Training Data R^2=',round(corr_train**2,4),'R=',round(corr_train,4))
```

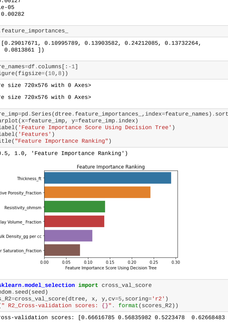
```
Training Data R^2= 0.9768 R= 0.9878
```

In [14]:
```
corr_test=np.corrcoef(y_test, y_pred_test) [0,1]
print('Testing Data R^2',round(corr_test**2,4),'R=',round(corr_test,4))
```

```
Testing Data R^2 0.8738 R= 0.8209
```

In [15]:
```
plt.figure(figsize=(10,8))
plt.plot(y_train, y_pred_train, 'g.')
plt.xlabel('TOC Training Actual')
plt.ylabel('TOC Training Prediction')
plt.title('TOC Training Actual Vs. Prediction')
```

Out[15]:
```
Text(0.5, 1.0, 'TOC Training Actual Vs. Prediction')
```



In [16]:
```
plt.figure(figsize=(10,8))
plt.plot(y_test, y_pred_test, 'g.')
plt.xlabel('TOC Testing Actual')
plt.ylabel('TOC Testing Prediction')
plt.title('TOC Testing Actual Vs. Prediction')
```

Out[16]:
```
Text(0.5, 1.0, 'TOC Testing Actual Vs. Prediction')
```


In [17]:
```
TOC_Actual_Prediction=pd.DataFrame({'Actual':y_test,'Predicted':y_pred_test})
TOC_Actual_Prediction
```

Out[17]:

	Actual	Predicted
834	0.051979	0.051979
604	0.053110	0.053445
747	0.057635	0.056708
908	0.057235	0.054457
545	0.054575	0.054701
...		
808	0.056807	0.056949
166	0.054616	0.054362
172	0.058651	0.058570
263	0.058120	0.057927
427	0.051019	0.049853

297 rows × 2 columns

In [18]:
```
from sklearn import metrics
print('MAE:', round(metrics.mean_absolute_error(y_test,y_pred_test),5))
print('MSE:', round(metrics.mean_squared_error(y_test,y_pred_test),5))
print('RMSE:', round(np.sqrt(metrics.mean_squared_error(y_test,y_pred_test)),5))
```

```
MAE: 0.00127
MSE: 1e-05
RMSE: 0.00282
```

In [19]:
```
dtree.feature_importances_
```

Out[19]:
```
array([0.29017671, 0.10995789, 0.13903582, 0.24212085, 0.13732264,
       0.0013861 ])
```

In [20]:
```
feature_names=df.columns[:-1]
plt.figure(figsize=(10,5))
```

Out[20]:
```
<Figure size 720x576 with 0 Axes>
```

```
<Figure size 720x576 with 0 Axes>
```

In [21]:
```
feature_imp=pd.Series(dtree.feature_importances_,index=feature_names).sort_values(ascending=False)
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score Using Decision Tree')
plt.ylabel('Features')
plt.title("Feature Importance Ranking")
```

Out[21]:
```
Text(0.5, 1.0, 'Feature Importance Ranking')
```


In [22]:
```
from sklearn.model_selection import cross_val_score
np.random.seed(seed)
scores_R2=cross_val_score(dtree, x, y,cv=5,scoring='r2')
print('  R2_Cross-validation scores: {}'. format(scores_R2))
```

```
  R2_Cross-validation scores: [0.66616705 0.56835982 0.5223478  0.62668483 0.77734923]
```

In [23]:
```
print(' Average R2_Cross-validation scores: {}'.
format(scores_R2.mean()))
```

```
 Average R2_Cross-validation scores: 0.6321819058443482
```

In [ ]: