# crop

June 27, 2024

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
import pandas as pd
```

[4]:
```
df = pd.read_csv('/content/drive/MyDrive/archive (2)/data.csv')
```

[5]:
```
x = df[['moisture','temp']]
y = df['pump']
```

[6]:
```
k = 3
knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(x,y)
```

[6]:
```
KNeighborsClassifier(n_neighbors=3)
```

[7]:
```
new_data = np.array([[638,16]])
prediction = knn.predict(new_data)
if prediction == 1:
    print('pump')
else:
    print('not pump')
```

```
pump

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```

[11]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import KNeighborsClassifier
```

[13]:
```
new_data = np.array([[271, 33]])
prediction = knn.predict(new_data) # Use 'knn' instead of 'model'
print("Prediction for new data:", prediction)
```

```
Prediction for new data: [0]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```

[19]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪accuracy_score
```

[22]:
```
df = pd.read_csv('/content/drive/MyDrive/archive (2)/data.csv')
```

[23]:
```
print(df.head())
```

```
     crop  moisture  temp  pump
0  cotton       638    16     1
1  cotton       522    18     1
2  cotton       741    22     1
3  cotton       798    32     1
4  cotton       690    28     1
```

[24]:
```
X = df[['moisture', 'temp']]
y = df['pump']
```

[25]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[26]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

[26]:
```
LogisticRegression()
```

[28]:
```
from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        31

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

[27]:
```
y_pred = model.predict(X_test)
```

[29]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

[31]:
```
df = pd.read_csv('/content/drive/MyDrive/archive (2)/data.csv')
```

[32]:
```
print(df.head())
```

```
     crop  moisture  temp  pump
0  cotton       638    16     1
1  cotton       522    18     1
2  cotton       741    22     1
3  cotton       798    32     1
4  cotton       690    28     1
```

[33]:
```
X = df[['moisture', 'temp']]
y = df['pump']
```

[34]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

[35]:
```
dt_regressor = DecisionTreeRegressor(random_state=42)
```

[36]:
```
dt_regressor.fit(X_train, y_train)
DecisionTreeRegressor(random_state=42)
```

[36]:
```
DecisionTreeRegressor(random_state=42)
```

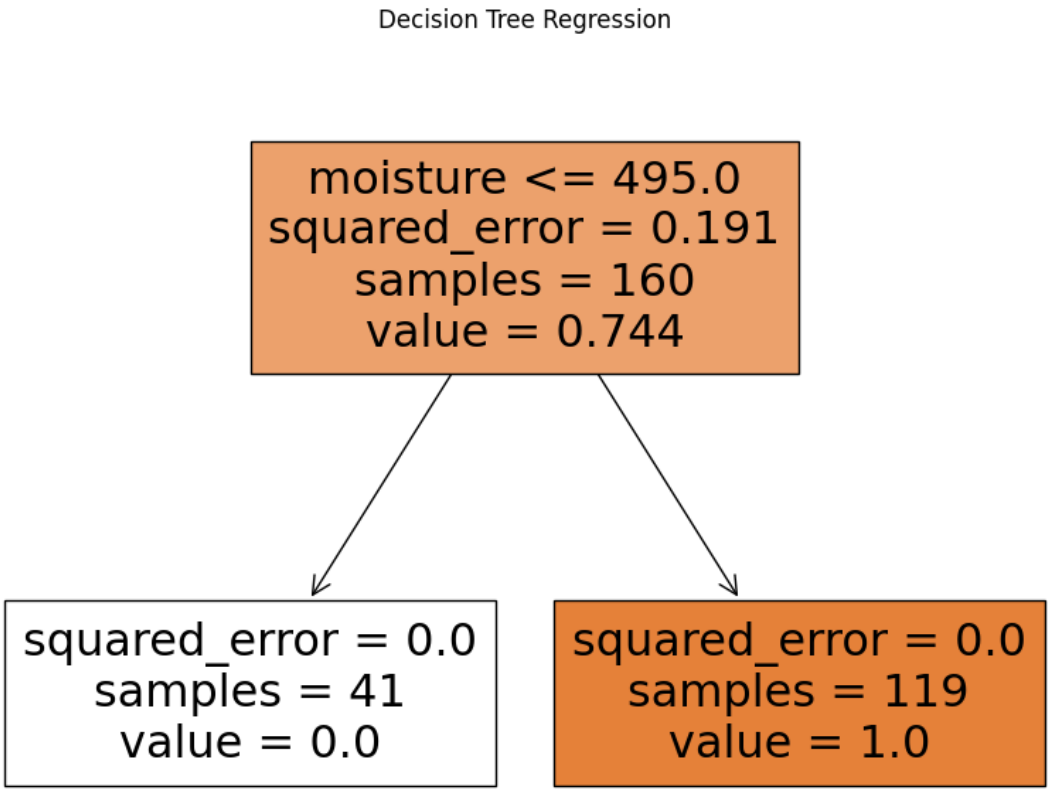
[37]:
```
y_pred = dt_regressor.predict(X_test)
```

[38]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[39]:
```
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

```
Mean Squared Error (MSE): 0.0
R-squared (R2): 1.0
```

[43]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(10, 8))
plot_tree(dt_regressor, feature_names=X.columns, filled=True)
plt.title("Decision Tree Regression")
plt.show()
```



[44]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

[45]:
```
df = pd.read_csv('/content/drive/MyDrive/archive (2)/data.csv')
```

[46]:
```
X = df[['moisture', 'temp']]
y = df['pump']
```

[47]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

[48]:
```
rf_regressor = RandomForestRegressor(n_estimators=100,
random_state=42)
```

[49]:
```
rf_regressor.fit(X_train, y_train)
RandomForestRegressor(random_state=42)
```

[49]:
```
RandomForestRegressor(random_state=42)
```

[50]:
```
y_pred = rf_regressor.predict(X_test)
```

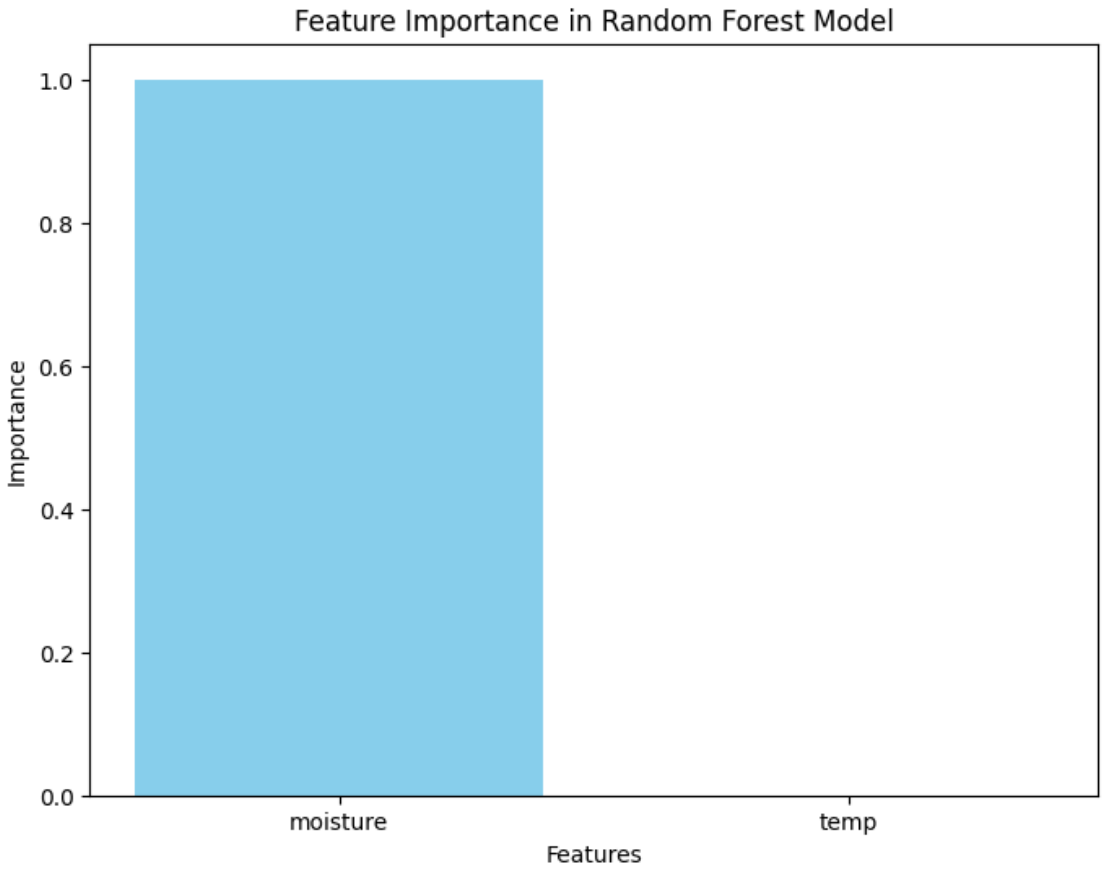
[51]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

[53]:
```
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

```
Mean Squared Error (MSE): 0.0014399999999999999
R-squared (R2): 0.991741935483871
```

[54]:
```
feature_importances = rf_regressor.feature_importances_
features = X.columns
```

[55]:
```
plt.figure(figsize=(8, 6))
plt.bar(features, feature_importances, color='skyblue')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Random Forest Model')
plt.show()
```




Spotify tracker

[ ]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
import pandas as pd
```

[ ]:
```
df = pd.read_csv('/content/drive/MyDrive/archive (3)/dataset.csv')
```

[ ]:
```
x = df[['danceability','energy']]
y = df['mode']
```

[ ]:
```
k = 3
knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(x,y)
```

[ ]:
```
KNeighborsClassifier(n_neighbors=3)
```

[ ]:
```
new_data = np.array([[0.755,0.454]])
prediction = knn.predict(new_data)
if prediction == 1:
    print('mode')
else:
    print('not mode')
```

```
not mode

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```

[ ]:
```
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
```

[ ]:
```
from sklearn.model_selection import train_test_split as ttp
from sklearn.metrics import classification_report
```

[ ]:
```
y_numeric = y.replace({'yes': 1, 'no': 0})

# Now fit the model with the numeric target variable
LR.fit(x, y_numeric)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
new_data = np.array([[0.755, 0.454]])
prediction = LR.predict(new_data)[0]

ph_value = new_data[0][0]

# Check if the prediction indicates the crop can be grown
if ph_value > 5:
    print('mode')
else:
    print('not mode')
```

```
not mode

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```