task3

April 29, 2025

[6]:
```python
from ucimlrepo import fetch_ucirepo

# fetch dataset
bank_marketing = fetch_ucirepo(id=222)

# data (as pandas dataframes)
X = bank_marketing.data.features
y = bank_marketing.data.targets

# metadata
print(bank_marketing.metadata)

# variable information
print(bank_marketing.variables)
```


```
{'uci_id': 222, 'name': 'Bank Marketing', 'repository_url':
'https://archive.ics.uci.edu/dataset/222/bank+marketing', 'data_url':
'https://archive.ics.uci.edu/static/public/222/data.csv', 'abstract': 'The data
is related with direct marketing campaigns (phone calls) of a Portuguese banking
institution. The classification goal is to predict if the client will subscribe
a term deposit (variable y).', 'area': 'Business', 'tasks': ['Classification'],
'characteristics': ['Multivariate'], 'num_instances': 45211, 'num_features': 16,
'feature_types': ['Categorical', 'Integer'], 'demographics': ['Age',
'Occupation', 'Marital Status', 'Education Level'], 'target_col': ['y'],
'index_col': None, 'has_missing_values': 'yes', 'missing_values_symbol': 'NaN',
'year_of_dataset_creation': 2014, 'last_updated': 'Fri Aug 18 2023',
'dataset_doi': '10.24432/C5K306', 'creators': ['S. Moro', 'P. Rita', 'P.
Cortez'], 'intro_paper': {'ID': 277, 'type': 'NATIVE', 'title': 'A data-driven
approach to predict the success of bank telemarketing', 'authors': 'Sérgio Moro,
P. Cortez, P. Rita', 'venue': 'Decision Support Systems', 'year': 2014,
'journal': None, 'DOI': '10.1016/j.dss.2014.03.001', 'URL': 'https://www.semanti
cscholar.org/paper/cab86052882d126d43f72108c6cb41b295cc8a9e', 'sha': None,
'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': None, 'pmcid':
None}, 'additional_info': {'summary': "The data is related with direct marketing
campaigns of a Portuguese banking institution. The marketing campaigns were
based on phone calls. Often, more than one contact to the same client was
required, in order to access if the product (bank term deposit) would be ('yes')
or not ('no') subscribed. \n\nThere are four datasets: \n1) bank-additional-
```

full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]\n2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.\n3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs). \n4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs). \nThe smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM). \n\nThe classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).", 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Input variables:\n # bank client data:\n   1 - age (numeric)\n   2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student",\n   "blue-collar","self-employed","retired","technician","services") \n   3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)\n   4 - education (categorical: "unknown","secondary","primary","tertiary")\n   5 - default: has credit in default? (binary: "yes","no")\n   6 - balance: average yearly balance, in euros (numeric) \n   7 - housing: has housing loan? (binary: "yes","no")\n   8 - loan: has personal loan? (binary: "yes","no")\n   # related with the last contact of the current campaign:\n   9 - contact: contact communication type (categorical: "unknown","telephone","cellular") \n 10 - day: last contact day of the month (numeric)\n 11 - month: last contact month of year (categorical: "jan", "feb", "mar", …, "nov", "dec")\n 12 - duration: last contact duration, in seconds (numeric)\n   # other attributes:\n 13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)\n 14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)\n 15 - previous: number of contacts performed before this campaign and for this client (numeric)\n 16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")\n\n Output variable (desired target):\n 17 - y - has the client subscribed a term deposit? (binary: "yes","no")\n', 'citation': None}}

```
           name     role         type      demographic  \
0           age  Feature      Integer              Age
1           job  Feature  Categorical       Occupation
2       marital  Feature  Categorical   Marital Status
3     education  Feature  Categorical  Education Level
4       default  Feature       Binary             None
5       balance  Feature      Integer             None
6       housing  Feature       Binary             None
7          loan  Feature       Binary             None
8       contact  Feature  Categorical             None
9   day_of_week  Feature         Date             None
10        month  Feature         Date             None
11     duration  Feature      Integer             None
```

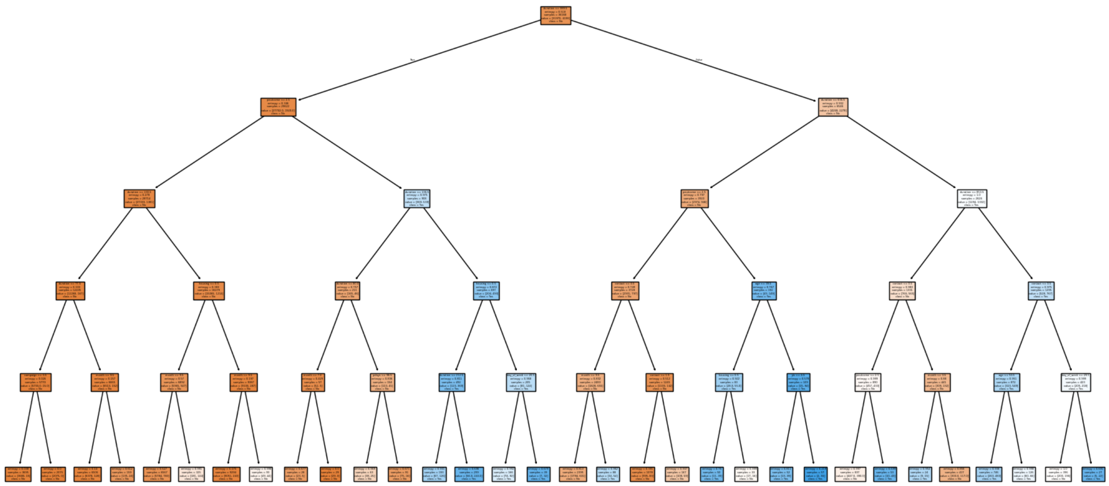
```
12     campaign  Feature      Integer           None
13        pdays  Feature      Integer           None
14     previous  Feature      Integer           None
15     poutcome  Feature  Categorical           None
16            y   Target       Binary           None

                                        description  units missing_values
0                                              None   None             no
1   type of job (categorical: 'admin.','blue-colla…   None             no
2   marital status (categorical: 'divorced','marri…   None             no
3   (categorical: 'basic.4y','basic.6y','basic.9y'…   None             no
4                             has credit in default?   None             no
5                             average yearly balance  euros             no
6                                  has housing loan?   None             no
7                                 has personal loan?   None             no
8   contact communication type (categorical: 'cell…   None            yes
9                       last contact day of the week   None             no
10  last contact month of year (categorical: 'jan'…   None             no
11   last contact duration, in seconds (numeric). …   None             no
12  number of contacts performed during this campa…   None             no
13  number of days that passed by after the client…   None            yes
14  number of contacts performed before this campa…   None             no
15  outcome of the previous marketing campaign (ca…   None            yes
16          has the client subscribed a term deposit?   None             no
```

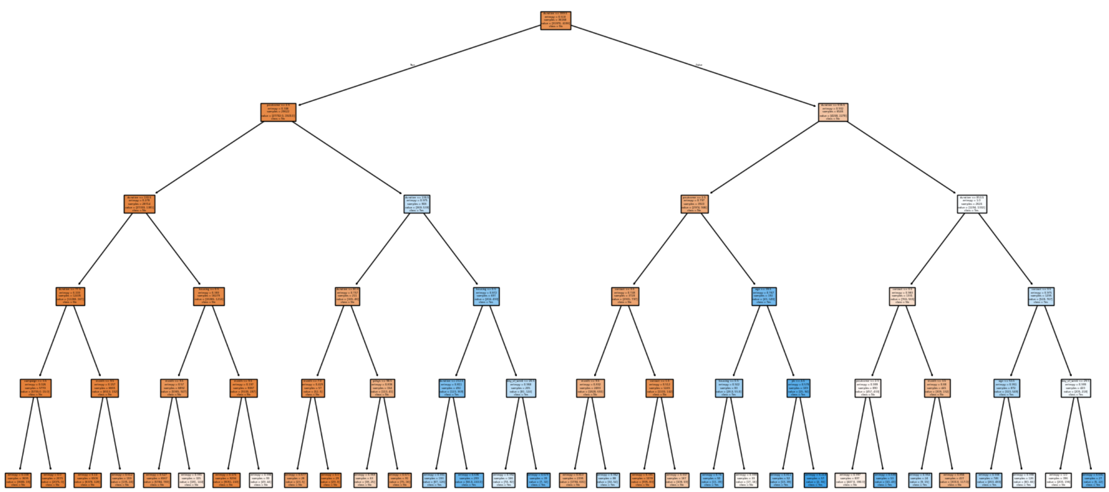
[9]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report , accuracy_score,␣
↪confusion_matrix


# Encode all categorical features
X_encoded = X.copy()
label_encoders = {}

for col in X_encoded.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X_encoded[col] = le.fit_transform(X_encoded[col].astype(str))
    label_encoders[col] = le

# Encode the target variable
y_encoded = LabelEncoder().fit_transform(y.values.ravel())

X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded,␣
↪test_size=0.2, random_state=42)
```

```python
model = DecisionTreeClassifier(criterion='entropy', max_depth=5,␣
 ↪random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
plot_tree(model, filled=True, feature_names=X.columns, class_names=["No",␣
 ↪"Yes"])
plt.show()


from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
plot_tree(model, filled=True, feature_names=X.columns, class_names=["No",␣
 ↪"Yes"])
plt.show()
```

```
Accuracy: 0.8947252018135574
Confusion Matrix:
 [[7778  174]
 [ 778  313]]
Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.98      0.94      7952
           1       0.64      0.29      0.40      1091

    accuracy                           0.89      9043
   macro avg       0.78      0.63      0.67      9043
weighted avg       0.88      0.89      0.88      9043
```





[10]:

```
#Different max_depth Values

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

for depth in range(1, 11):
    model = DecisionTreeClassifier(max_depth=depth, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Depth: {depth}, Accuracy: {accuracy_score(y_test, y_pred):.4f}")
```

```
Depth: 1, Accuracy: 0.8794
Depth: 2, Accuracy: 0.8932
Depth: 3, Accuracy: 0.8965
Depth: 4, Accuracy: 0.8965
Depth: 5, Accuracy: 0.8959
Depth: 6, Accuracy: 0.8988
Depth: 7, Accuracy: 0.8984
Depth: 8, Accuracy: 0.9021
Depth: 9, Accuracy: 0.9000
Depth: 10, Accuracy: 0.8997
```

[13]:
```python
#Test with Other Models

#Random Forest:

from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds))
```

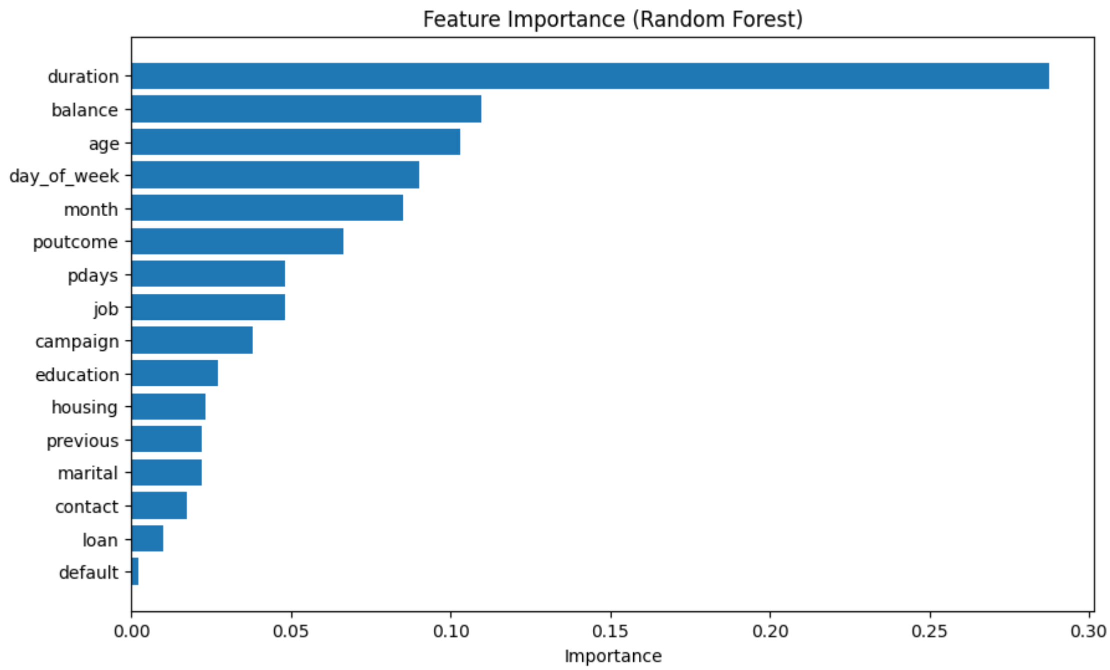
```
Random Forest Accuracy: 0.9022448302554462
```

[14]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_scaled, y_train)
lr_preds = lr_model.predict(X_test_scaled)
print("Logistic Regression Accuracy:", accuracy_score(y_test, lr_preds))
```

```
Logistic Regression Accuracy: 0.8942828707287405
```

[15]:
```python
#Analyze Feature Importance (for Tree-Based Models)
import pandas as pd
import matplotlib.pyplot as plt

importances = rf_model.feature_importances_
features = X_train.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Plotting
```

```
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'])
plt.xlabel("Importance")
plt.title("Feature Importance (Random Forest)")
plt.gca().invert_yaxis()
plt.show()
```



[ ]:

```
```