Mohammed Meraj - T2 - 32

Dataset : Employee Records Dataset

Lab 7 : Implement Random Forest algorithm

Importing the required libraries

In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [20]:
```
df = pd.read_csv('employee_records_dataset.csv')
```

In [21]:
```
df.head()
```

Out[21]:

	experience_years	remote	projects_completed	monthly_hours	promotion_last_year
0	38	0	12	202	0
1	28	0	19	118	0
2	14	0	10	125	1
3	7	0	9	108	0
4	20	1	5	269	1

In [22]:
```
df.shape
```

Out[22]:
```
(2500, 5)
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['experience_years', 'remote', 'projects_completed', 'monthly_hours',
       'promotion_last_year'],
      dtype='object')
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   experience_years     2500 non-null   int64
 1   remote               2500 non-null   int64
 2   projects_completed   2500 non-null   int64
 3   monthly_hours        2500 non-null   int64
 4   promotion_last_year  2500 non-null   int64
dtypes: int64(5)
memory usage: 97.8 KB
```

In [25]:
```
df.describe()
```

Out[25]:

	experience_years	remote	projects_completed	monthly_hours	promotion_last_year
count	2500.000000	2500.000000	2500.000000	2500.000000	2500.000000
mean	20.275600	0.475600	10.577600	189.399200	0.488800
std	11.826394	0.499504	5.762539	63.531315	0.499975
min	0.000000	0.000000	1.000000	80.000000	0.000000
25%	10.000000	0.000000	6.000000	136.000000	0.000000
50%	21.000000	0.000000	11.000000	189.000000	0.000000
75%	30.000000	1.000000	16.000000	243.000000	1.000000
max	40.000000	1.000000	20.000000	300.000000	1.000000

Putting Feature Variable to X and Target variable to y.

In [26]:
```
# Putting feature variable to X
X = df.drop('promotion_last_year',axis=1)
# Putting response variable to y
y = df['promotion_last_year']
```

Train-Test-Split is performed

In [27]:
```
# now lets split the data into train and test
from sklearn.model_selection import train_test_split
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=42)
X_train.shape, X_test.shape
```

Out[27]:
```
((1750, 4), (750, 4))
```

Let's import RandomForestClassifier and fit the data.

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1, max_depth=5,
                                       n_estimators=100, oob_score=True)
#%%time
classifier_rf.fit(X_train, y_train)
```

Out[28]:
```
RandomForestClassifier(max_depth=5, n_jobs=-1, oob_score=True, random_state=42)
```

In [39]:
```
y_pred = classifier_rf.predict(X_test)

# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print results
print(f"OOB Score:   {classifier_rf.oob_score_:.4f}")
print(f"Accuracy:    {accuracy:.4f}")
print(f"Precision:   {precision:.4f}")
print(f"Recall:      {recall:.4f}")
print(f"F1 Score:    {f1:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
OOB Score:   0.4691
Accuracy:    0.4813
Precision:   0.4607
Recall:      0.3342
F1 Score:    0.3874

Classification Report:
              precision    recall  f1-score   support

           0       0.49      0.62      0.55       382
           1       0.46      0.33      0.39       368

    accuracy                           0.48       750
   macro avg       0.48      0.48      0.47       750
weighted avg       0.48      0.48      0.47       750
```

In [29]:
```
# checking the oob score
classifier_rf.oob_score_
```

Out[29]:
```
0.46914285714285714
```

Let's do hyperparameter tuning for Random Forest using GridSearchCV and fit the data.

In [30]:
```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
from sklearn.model_selection import GridSearchCV
```

```
# Instantiate the grid search model
grid_search = GridSearchCV(estimator=rf,
                           param_grid=params,
                           cv = 4,
                           n_jobs= 5, verbose=1, scoring="accuracy")
#%%time
grid_search.fit(X_train, y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits
```

Out[30]:

GridSearchCV

best_estimator_: RandomForestClassifier

RandomForestClassifier

In [31]:
```
grid_search.best_score_
```

Out[31]:
```
np.float64(0.5000065306207747)
```

Task

computer f1 score, accuracy

In [32]:
```
rf_best = grid_search.best_estimator_
rf_best
```

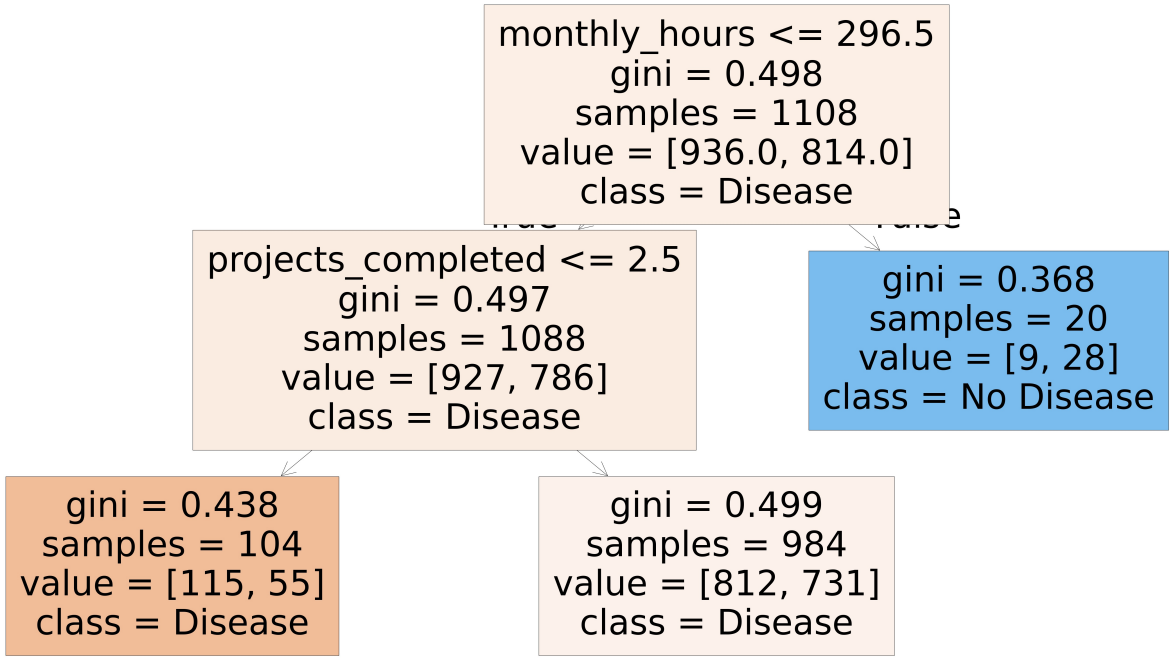
Out[32]:

RandomForestClassifier

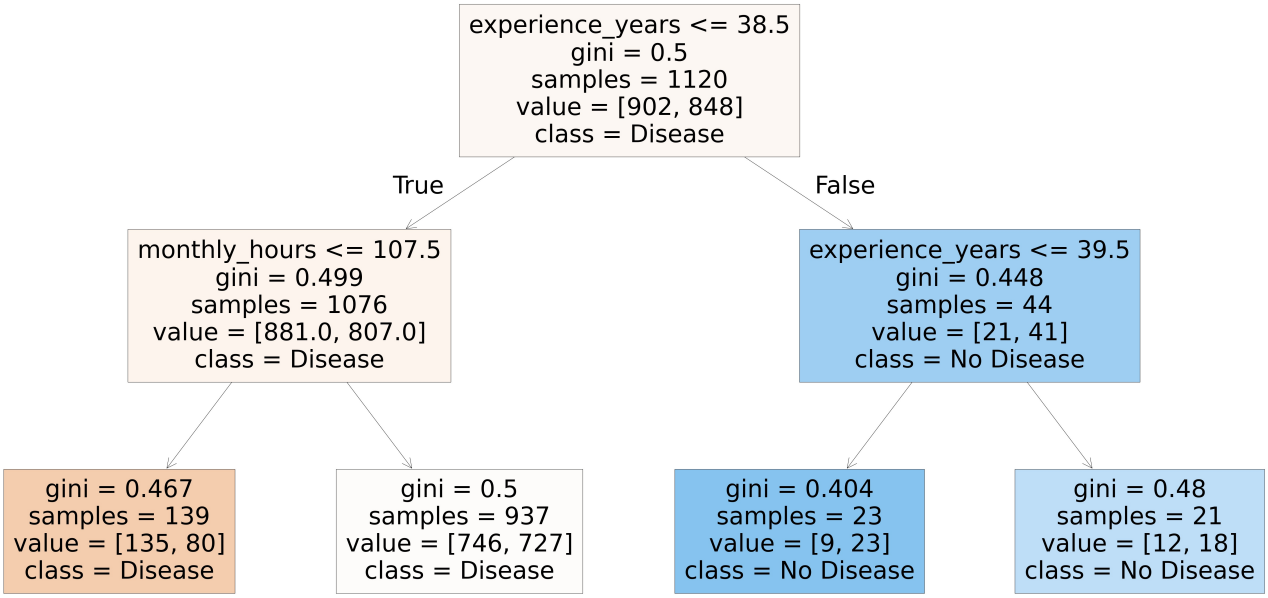
```
RandomForestClassifier(max_depth=2, min_samples_leaf=20, n_estimators=10,
                       n_jobs=-1, random_state=42)
```

Now, let's visualize

In [33]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5], feature_names = X.columns,class_names=['Disease', "No Disease"],filled=True);
```



In [34]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7], feature_names = X.columns,class_names=['Disease', "No Disease"],filled=True);
```



The trees created by estimators_[5] and estimators_[7] are different. Thus we can say that each tree is independent of the other.

Now let's sort the data with the help of feature importance

In [35]:
```
rf_best.feature_importances_
```

Out[35]:
```
array([0.32020311, 0.05150207, 0.29455841, 0.33373641])
```

In [36]:
```
## feature importance

imp_df = pd.DataFrame({
    "Varname": X_train.columns,
    "Imp": rf_best.feature_importances_
})
```

In [37]:
```
imp_df.sort_values(by="Imp", ascending=False)
```

Out[37]:

	Varname	Imp
3	monthly_hours	0.333736
0	experience_years	0.320203
2	projects_completed	0.294558
1	remote	0.051502

In [38]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
```

In [ ]: