Classification models

In this file we will train some classic classification models for comparisson with the BERT model we trained for the first screening of resumes.

We will be training a decision tree, and random forest model and a logistic regression classifier.

These are all lighter easier to load than the NLP BERT model we trained earlier, but can still yield decent results

In [9]:
```
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import classification_report
```

In [10]:
```
# Loading the pickled dataset
import pickle
with open('Data/Dataframes/newDF.pkl', 'rb') as f:
    df = pickle.load(f)
```

In [ ]:
```
# As we did in the previous notebook, we will drop the columns not used for trainin
trainingDF = df.drop(columns=['ID', 'Label', 'TextLen'])
```

In [12]:
```
# Make sure we have the correct columns
trainingDF.columns
```

Out[12]:
```
Index(['Resume', 'Software_Developer', 'Database_Administrator',
       'Systems_Administrator', 'Project_manager', 'Web_Developer',
       'Network_Administrator', 'Security_Analyst', 'Python_Developer',
       'Java_Developer', 'Front_End_Developer'],
      dtype='object')
```

In [13]:
```
X = trainingDF['Resume']
y = trainingDF.drop(columns=['Resume'])

# Splitting the dataset into training and testing sets. 10 % of the data will be us
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_sta
```

Vectorization of text data

We are again working with resume texts, which we cant just feed into any ML model.

First we have to vectorize it. We will use the TfidfVectorizer, which is common for these types of models.

In [ ]:
```
# We cap the number of features of the vector to 5000 to start with, but we might r
vectorizer = TfidfVectorizer(max_features=5000)
```

```
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

In [ ]:
```
# Function to evaluate the model
def evaluate_model(name, model, X_train, y_train, X_test, y_test, labels):
    print(f"\n{name}: ")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred, target_names=labels))
```

In [ ]:
```
# Training and evaluating the decision tree classifier
dt_model = MultiOutputClassifier(DecisionTreeClassifier(random_state=42))
evaluate_model("Decision Tree", dt_model, X_train_vec, y_train, X_test_vec, y_test,

# Training and evaluating the Logistic regression classifier
lr_model = MultiOutputClassifier(LogisticRegression(max_iter=1000))
evaluate_model("Logistic Regression", lr_model, X_train_vec, y_train, X_test_vec, y
```

```
=== Decision Tree ===
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in samples with no true nor predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
                      precision    recall  f1-score   support

    Software_Developer       0.94      0.94      0.94      1451
Database_Administrator       0.84      0.82      0.83       299
 Systems_Administrator       0.73      0.76      0.74       569
       Project_manager       0.72      0.76      0.74       437
         Web_Developer       0.74      0.75      0.75       605
 Network_Administrator       0.75      0.70      0.72       444
      Security_Analyst       0.79      0.80      0.79       314
      Python_Developer       0.90      0.85      0.87       269
        Java_Developer       0.87      0.85      0.86       329
   Front_End_Developer       0.84      0.85      0.85       387

             micro avg       0.83      0.83      0.83      5104
             macro avg       0.81      0.81      0.81      5104
          weighted avg       0.83      0.83      0.83      5104
           samples avg       0.81      0.82      0.79      5104


=== Logistic Regression ===
                      precision    recall  f1-score   support

    Software_Developer       0.97      0.94      0.96      1451
Database_Administrator       0.97      0.72      0.83       299
 Systems_Administrator       0.86      0.71      0.77       569
       Project_manager       0.90      0.70      0.79       437
         Web_Developer       0.81      0.72      0.76       605
 Network_Administrator       0.87      0.63      0.73       444
      Security_Analyst       0.90      0.77      0.83       314
      Python_Developer       0.95      0.78      0.86       269
        Java_Developer       0.94      0.78      0.85       329
   Front_End_Developer       0.95      0.80      0.87       387

             micro avg       0.92      0.79      0.85      5104
             macro avg       0.91      0.76      0.82      5104
          weighted avg       0.92      0.79      0.85      5104
           samples avg       0.86      0.79      0.81      5104
```

```
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in samples with no true nor predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Grid searching for the best parameters for the Random Forest Classifier

In our earlier projects we had good results with the RF classifier, but it sometimes requires tinkering with the parameters.

We will therefore search for the best candidates with the following function:

In [ ]:

```
# Reusable function to run GridSearchCV for Random Forest
# Define parameter grid for tuning
param_grid = {
    'estimator__n_estimators': [50, 100],
    'estimator__max_depth': [None, 10, 30],
    'estimator__min_samples_split': [2, 5],
    'estimator__min_samples_leaf': [1, 2]
}

    # Setup the base model
rf = MultiOutputClassifier(RandomForestClassifier(random_state=42))

    # Setup Grid Search
grid_search = GridSearchCV(
    rf,
    param_grid,
    cv=3,
    scoring='f1_micro',
    verbose=3,
    n_jobs=-1
)

    # Train the grid search
grid_search.fit(X_train_vec, y_train)

    # Print best parameters found
print("\nBest Parameters Found (Random Forest):")
print(grid_search.best_params_)

    # Evaluate the best model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_vec)

print("\nClassification Report for Optimized Random Forest:")
print(classification_report(y_test, y_pred, target_names=y.columns))
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

Best Parameters Found (Random Forest):
{'estimator__max_depth': None, 'estimator__min_samples_leaf': 1, 'estimator__min_s
amples_split': 5, 'estimator__n_estimators': 100}

Classification Report for Optimized Random Forest:
                        precision    recall  f1-score   support

    Software_Developer       0.97      0.95      0.96      1451
Database_Administrator       0.99      0.69      0.81       299
 Systems_Administrator       0.86      0.69      0.76       569
       Project_manager       0.95      0.64      0.77       437
         Web_Developer       0.82      0.72      0.76       605
 Network_Administrator       0.89      0.61      0.73       444
      Security_Analyst       0.92      0.65      0.76       314
      Python_Developer       0.99      0.74      0.85       269
        Java_Developer       0.93      0.81      0.87       329
   Front_End_Developer       0.95      0.81      0.88       387

             micro avg       0.93      0.77      0.84      5104
             macro avg       0.93      0.73      0.81      5104
          weighted avg       0.93      0.77      0.84      5104
           samples avg       0.85      0.77      0.79      5104
```

```
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\pelle\Work\1semSoft\exam\AIML-Exam\examVenv\Lib\site-packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in samples with no true nor predicted labels. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [ ]:
```python
import os
import pickle

# Make sure the directory exists
os.makedirs("models", exist_ok=True)

# Save vectorizer
with open("models/classificationModels/vectorizer.pkl", "wb") as f:
    pickle.dump(vectorizer, f)

# Save models
with open("models/classificationModels/decision_tree.pkl", "wb") as f:
    pickle.dump(dt_model, f)

with open("models/classificationModels/logistic_regression.pkl", "wb") as f:
    pickle.dump(lr_model, f)

with open("models/classificationModels/random_forest_best.pkl", "wb") as f:
    pickle.dump(grid_search.best_estimator_, f)
```

```
The Kernel crashed while executing code in the current cell or a previous cell.

Please review the code in the cell(s) to identify a possible cause of the failure.

Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more info.

View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.
```

Evaluation of the classic classification models

To compare with the BERT model we trained earlier for first-round resume screening, we tested three classic machine learning models: Decision Tree, Logistic Regression, and Random Forest. All models were trained using TF-IDF vectors based on the resume texts and evaluated using standard multi-label classification metrics.

Decision Tree

The Decision Tree model was quick to train and easy to interpret. It managed to score reasonably well on high-frequency labels like Software_Developer (F1-score: 0.94) and Python_Developer (0.87), but its performance dropped for several other categories such as Systems_Administrator (0.74) and Project_manager (0.74). The micro average F1-score ended at 0.83. This means that the model have a decent chance can label highly occuring labels like Software Developer instead of the actual label.

Logistic Regression

Logistic Regression performed decently above the board. It gave high precision and recall on the most common labels, and also handled the less frequent ones more consistently. For example, it achieved F1-scores of 0.96 for Software_Developer, 0.89 for Java_Developer, and 0.88 for Front_End_Developer. The micro average F1-score was 0.85, and the samples average F1-score landed at 0.81.

Random Forest (with GridSearch)

To get the best results out of the Random Forest model, we performed a grid search over 24 combinations of parameters, resulting in 72 total fits. The best-performing parameters were:

{ 'estimator**max_depth': None, 'estimator**min_samples_leaf': 1,
'estimator**min_samples_split': 5, 'estimator**n_estimators': 100 }

With these parameters, the Random Forest model achieved almost the same results, but slightly better. It reached an F1-score of 0.96 for Software_Developer, 0.87 for Java_Developer, and 0.88 for Front_End_Developer. While performance dropped slightly on some of the less represented classes, such as Database_Administrator (0.81) and Security_Analyst (0.76), the overall micro average F1-score was 0.84, and the samples average F1-score was 0.79. The Grid search was almost negligible for the performance, giving it 2-3 % more on the f1 score.

Comparing the models

Model	Micro F1	Macro F1	Weighted F1	Precision (Macro)	Recall (Macro)
Decision Tree	0.83	0.81	0.83	0.81	0.81
Logistic Regression	0.85	0.82	0.85	0.91	0.76
Random Forest	0.84	0.81	0.84	0.93	0.73

The models all scored almost the same, but they still have different pros and cons. Of the models we would might use the RF and Logistic Reg models in the future, but not the Decision Tree.

The decision tree has problems assessing the labels, since it can either be positive (100%) or negative (0%), which means that we cant sort the Resumes on a threshold designed by the HR department for the use case. The Random Forest model uses many Decision Trees and can give an actual score percentage, which we can use, just like the Logistic Regression Classifier.