resume_trained_model

October 17, 2025

[28]:
```
import pandas as pd
import numpy as np
import joblib
import os
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score,␣
 ↪log_loss
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.model_selection import RandomizedSearchCV
import torch
```

[29]:
```
# --- Configuration ---
RANDOM_SEED = 42
MODEL_DIR = 'models/'
os.makedirs(MODEL_DIR, exist_ok=True) # Ensure directory exists
EMBEDDING_MODEL_NAME = 'all-MiniLM-L6-v2'
print(f"Ensured directory '{MODEL_DIR}' exists.")
```

```
Ensured directory 'models/' exists.
```

[30]:
```
# --- Initialize Models (Run once) ---
print("Loading SBERT Model...")
SBERT_MODEL = SentenceTransformer(EMBEDDING_MODEL_NAME)
EMBEDDING_DIM = SBERT_MODEL.get_sentence_embedding_dimension()
print(f"SBERT Model Loaded. Embedding Dimension: {EMBEDDING_DIM}")
```

```
Loading SBERT Model…
SBERT Model Loaded. Embedding Dimension: 384
```

[31]:
```
# --- Hybrid Feature Extraction ---

def get_semantic_embedding(text):
    """Generates the semantic embedding vector using SBERT."""
    # SBERT can process lists, which is more efficient
    if isinstance(text, str):
        text = [text]
```

```
    return SBERT_MODEL.encode(text, convert_to_numpy=True)
```

[32]:
```
def combine_hybrid_features(df, fitted_tfidf, sbert_model):
    """
    Transforms the JD-Resume pairs into the hybrid numerical feature matrix (X).
    X_features will have columns: [TFIDF_Similarity, SBERT_Similarity]
    """
    X_features = []

    # Get embeddings for all JD and Resume texts in one batch (much faster)
    all_jd_emb = sbert_model.encode(df['job_description'].tolist(),␣
 ↪convert_to_numpy=True)
    all_res_emb = sbert_model.encode(df['resume_text'].tolist(),␣
 ↪convert_to_numpy=True)

    # Get TF-IDF vectors for all texts in one batch
    all_jd_tfidf = fitted_tfidf.transform(df['job_description'].tolist())
    all_res_tfidf = fitted_tfidf.transform(df['resume_text'].tolist())

    for i in range(len(df)):
        # 1. Lexical Feature (TF-IDF Similarity)
        # Cosine similarity between JD and Resume TF-IDF vectors
        tfidf_similarity = cosine_similarity(all_jd_tfidf[i],␣
 ↪all_res_tfidf[i])[0][0]

        # 2. Semantic Feature (SBERT Similarity)
        # Cosine similarity between JD and Resume SBERT embeddings
        # Reshape is needed for cosine_similarity function if only one vector
        semantic_similarity = cosine_similarity(
            all_jd_emb[i].reshape(1, -1),
            all_res_emb[i].reshape(1, -1)
        )[0][0]

        # 3. Concatenate the final feature vector for XGBoost
        feature_vector = [tfidf_similarity, semantic_similarity]
        X_features.append(feature_vector)

    return np.array(X_features)
```

[33]:
```
# Load the new data
df = pd.read_csv("job_resume_pairs.csv")

# --- Use the explicit columns ---
# X_text now contains the two text columns
X_text = df[['job_description', 'resume_text']]
# y is the explicit label
y = df['label']
```

```
print(f"Dataset Size: {len(df)} samples")
print(f"Fit (1) examples: {df['label'].sum()}")
print(f"No Fit (0) examples: {len(df) - df['label'].sum()}")
```

```
Dataset Size: 590 samples
Fit (1) examples: 295
No Fit (0) examples: 295
```

[34]:
```
# 1. Split into Training Pool (80%) and Final Test Set (20%)
X_pool, X_test, y_pool, y_test = train_test_split(
    X_text, y, test_size=0.2, random_state=RANDOM_SEED, stratify=y
)

# 2. Split Training Pool into Training Set (75% of pool) and Validation Set␣
 ↪(25% of pool)
X_train, X_val, y_train, y_val = train_test_split(
    X_pool, y_pool, test_size=(0.25), random_state=RANDOM_SEED, stratify=y_pool
)

print("-" * 30)
print(f"Training Set Size: {len(X_train)}")
print(f"Validation Set Size: {len(X_val)}")
print(f"Test Set Size: {len(X_test)}")
print("-" * 30)
```

```
------------------------------
Training Set Size: 354
Validation Set Size: 118
Test Set Size: 118
------------------------------
```

[35]:
```
# --- 4. Feature Transformation and XGBoost Training (Tuned) ---

# --- STEP 1: TF-IDF & FEATURE EXTRACTION ---
# (This part remains the same)
print("Fitting TF-IDF Vectorizer on ALL Training Text...")
train_corpus = X_train['job_description'].tolist() + X_train['resume_text'].
 ↪tolist()
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_vectorizer.fit(train_corpus)
joblib.dump(tfidf_vectorizer, MODEL_DIR + 'tfidf_vectorizer.pkl')

print("Extracting Hybrid Features...")
X_train_hybrid = combine_hybrid_features(X_train, tfidf_vectorizer, SBERT_MODEL)
X_test_hybrid = combine_hybrid_features(X_test, tfidf_vectorizer, SBERT_MODEL)
# NOTE: X_val_hybrid is NOT needed here; it is implicitly handled by CV on␣
 ↪X_train_hybrid.
```

```
# --- STEP 2: HYPERPARAMETER TUNING USING RANDOMIZEDSEARCHCV ---

print("-" * 40)
print("Starting Randomized Search for Optimal XGBoost Parameters...")
# 1. Define the parameter grid to search
param_grid = {
    # Number of trees (already set high, but can be tuned)
    'n_estimators': [500, 1000, 1500],
    # Maximum depth of a tree (Controls complexity)
    'max_depth': [3, 5, 7, 9],
    # Learning rate (How fast the model learns)
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    # Minimum loss reduction required to make a further partition (Controls␣
 ↪greediness)
    'gamma': [0, 0.1, 0.5],
    # Subsample ratio of the training instances (Reduces variance)
    'subsample': [0.6, 0.8, 1.0],
}

# 2. Initialize the base classifier
base_xgb = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss', # Log loss is the standard for binary classification
    random_state=RANDOM_SEED
)

# 3. Initialize RandomizedSearchCV
# n_iter=10 means it will test 10 random combinations from the grid.
# cv=3 means 3-fold cross-validation is performed on the training set for each␣
 ↪combination.
# scoring='f1' means the objective is to maximize the F1-Score (best for thesis/
 ↪imbalanced data).
random_search = RandomizedSearchCV(
    estimator=base_xgb,
    param_distributions=param_grid,
    n_iter=10,
    scoring='f1',
    cv=3,
    verbose=1,
    random_state=RANDOM_SEED,
    n_jobs=-1 # Use all available CPU cores for speed
)

# 4. Fit the search (This replaces the simple xgb_model.fit)
```

```python
# The search uses the Training Set and performs its own internal validation/
    ↪tuning.
random_search.fit(X_train_hybrid, y_train)

# --- STEP 3: FINAL MODEL SELECTION AND SAVING ---

# Retrieve the best model found during the search
optimal_xgb_model = random_search.best_estimator_

print("\nHyperparameter Tuning Complete.")
print(f"Best F1-Score found: {random_search.best_score_:.4f}")
print(f"Best Parameters: {random_search.best_params_}")

# Save the optimal model
joblib.dump(optimal_xgb_model, MODEL_DIR + 'xgb_classifier_optimal.pkl')
print(f"Optimal XGBoost Model saved to {MODEL_DIR}...")
```

```
Fitting TF-IDF Vectorizer on ALL Training Text…
Extracting Hybrid Features…
----------------------------------------
Starting Randomized Search for Optimal XGBoost Parameters…
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

```
  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
```

```
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/home/ferpaolo/tf-env/lib/python3.11/site-packages/xgboost/training.py:183:
UserWarning: [16:27:21] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)


Hyperparameter Tuning Complete.
Best F1-Score found: 0.9037
Best Parameters: {'subsample': 0.6, 'n_estimators': 1500, 'max_depth': 5,
'learning_rate': 0.05, 'gamma': 0.1}
Optimal XGBoost Model saved to models/…
```

[36]:
```
# --- 5. Final Model Evaluation ---

# The optimal model is stored in the best_estimator_ attribute of the search␣
 ↪object
# We use 'optimal_xgb_model' which was defined as 'random_search.
 ↪best_estimator_'
```

```python
optimal_xgb_model = random_search.best_estimator_


print("\n" + "="*40)
print("  FINAL MODEL EVALUATION (TEST SET)")
print("="*40)

# Predict on the unseen Test Set using the OPTIMAL model
y_pred = optimal_xgb_model.predict(X_test_hybrid)
y_proba = optimal_xgb_model.predict_proba(X_test_hybrid)[:, 1]

# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
logloss = log_loss(y_test, y_proba)


print(f"Optimal Model Accuracy on Test Set: {accuracy:.4f}")
print(f"Optimal Model F1-Score on Test Set: {f1:.4f}")
print(f"Optimal Model Log Loss on Test Set: {logloss:.4f}")

# Detailed report for thesis (Provides Precision, Recall, F1 for both classes)
print("\nClassification Report (ISO-IEC 25010 Metrics):")
print(classification_report(y_test, y_pred, target_names=['No Fit (0)', 'Fit␣
↪(1)']))
```

```
========================================
  FINAL MODEL EVALUATION (TEST SET)
========================================
Optimal Model Accuracy on Test Set: 0.9576
Optimal Model F1-Score on Test Set: 0.9573
Optimal Model Log Loss on Test Set: 0.1407

Classification Report (ISO-IEC 25010 Metrics):
              precision    recall  f1-score   support

  No Fit (0)       0.95      0.97      0.96        59
     Fit (1)       0.97      0.95      0.96        59

    accuracy                           0.96       118
   macro avg       0.96      0.96      0.96       118
weighted avg       0.96      0.96      0.96       118
```