XGBoost Model for Predicting Staff Termination

Author: Omkar Masurekar

Description: This notebook demonstrates the process of feature engineering, model training, and prediction using the XGBoost algorithm to identify the likelihood of staff termination. The workflow includes data preprocessing, handling categorical variables, and evaluating model performance to provide actionable insights into employee attrition risk.

In [392]:

```
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
```

In [393]:

```
df= pd.read_csv(r"./cleaned_test_csv.csv")
```

In [394]:

```
df
```

Out[394]:

	Unnamed: 0	FirstName	LastName	StartDate	ExitDate	PositionTitle	Supervisor	
0	0	Aaden	Mercer	2023-07-26	NaN	Senior Research Fellow	Victoria Jacobs	aaden
1	1	Aaliyah	Watts	2020-05-09	NaN	Senior Lecturer	Jared Whitehead	aaliya
2	2	Aarav	Espinoza	2019-11-11	NaN	Officer	David Ali	aarav.e
3	3	Aaron	Tapia	2023-01-20	NaN	Officer	Michael Reed	aar
4	4	Aaron	Weber	2020-04-06	2023-07-24	Lecturer	Melanie Garcia	aaro
...	...	...	...	...	...	...	...	
2995	2995	Zoey	Page	2019-05-06	2022-04-09	Professor	William Newman	zo
2996	2996	Zoey	Spence	2020-05-06	NaN	Senior Lecturer	Ian Price	zoey
2997	2997	Zoie	Logan	2019-04-19	NaN	Lecturer	Heidi Terry	zo
2998	2998	Zoie	Mercado	2020-12-01	NaN	Lecturer	David Young	zoie.n
2999	2999	Zoie	Rowland	2022-02-08	2022-07-22	Senior Lecturer	Mrs. Rachel Henry	zoie.

3000 rows × 42 columns

In [395]:

```python
df.columns
```

Out[395]:

```
Index(['Unnamed: 0', 'FirstName', 'LastName', 'StartDate', 'ExitDate',
       'PositionTitle', 'Supervisor', 'ADEmail', 'Faculty', 'EmployeeStatu
s',
       'EmployeeType', 'TerminationType', 'DOB', 'JobFunction', 'GenderCod
e',
       'Current Employee Rating', 'Employee ID', 'Survey Date',
       'Engagement Score', 'Satisfaction Score', 'Work-Life Balance Scor
e',
       'StaffType', 'EmploymentDuration_months', 'TerminationYear',
       'TerminationMonth', 'TerminationYearMonth', 'TenureGroup',
       'EmploymentStatus', 'StatusValidated', 'TerminationCategory',
       'VoluntaryTermination', 'InvoluntaryTermination', 'ActiveStaff',
       'RetirementCategory', 'TerminatedStaff', 'RetainedOver2Years',
       'Total_Survey_Score', 'PerformanceGroup', 'EngagementGroup',
       'SatisfactionGroup', 'Work-Life_BalanceGroup', 'Total_Survey_Grou
p'],
      dtype='object')
```

In [396]:

```python
df['DOB'] = pd.to_datetime(df['DOB'], dayfirst=True, errors='coerce')
df['StartDate'] = pd.to_datetime(df['StartDate'], errors='coerce')
df['Age'] = (pd.Timestamp.today() - df['DOB']).dt.days // 365
df['TerminationMonth'] = df['TerminationMonth'].fillna('Active')
```

In [397]:

```python
df['TenureGroup'].unique()
```

Out[397]:

```
array(['1-3yrs', '5-10yrs', '3-5yrs', '<1yr', '0'], dtype=object)
```

Since JobFunction and PositionTitle have a large number of categorical values, we applied label encoding to convert each category into an integer value. This approach helps simplify these features for the model and improves training efficiency.

In [398]:

```python
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline


for col in ['JobFunction', 'PositionTitle']:
    le = LabelEncoder()
    df[col] = df[col].fillna('Unknown')  # Handle missing values
    df[col + '_LE'] = le.fit_transform(df[col])
```

In [399]:

```
tenure_order = ['0', '<1yr', '1-3yrs', '3-5yrs', '5-10yrs', '10+yrs']

from sklearn.preprocessing import OrdinalEncoder

enc = OrdinalEncoder(categories=[tenure_order], dtype=int)
df['TenureGroup_'] = enc.fit_transform(df[['TenureGroup']])

```

In [400]:

```
df.groupby(['TenureGroup', 'TenureGroup_']).size()
```

Out[400]:

```
TenureGroup  TenureGroup_
0            0                  6
1-3yrs       2               1020
3-5yrs       3                725
5-10yrs      4                538
<1yr         1                711
dtype: int64
```

In [401]:

```
df.columns
```

Out[401]:

```
Index(['Unnamed: 0', 'FirstName', 'LastName', 'StartDate', 'ExitDate',
       'PositionTitle', 'Supervisor', 'ADEmail', 'Faculty', 'EmployeeStatu
s',
       'EmployeeType', 'TerminationType', 'DOB', 'JobFunction', 'GenderCod
e',
       'Current Employee Rating', 'Employee ID', 'Survey Date',
       'Engagement Score', 'Satisfaction Score', 'Work-Life Balance Scor
e',
       'StaffType', 'EmploymentDuration_months', 'TerminationYear',
       'TerminationMonth', 'TerminationYearMonth', 'TenureGroup',
       'EmploymentStatus', 'StatusValidated', 'TerminationCategory',
       'VoluntaryTermination', 'InvoluntaryTermination', 'ActiveStaff',
       'RetirementCategory', 'TerminatedStaff', 'RetainedOver2Years',
       'Total_Survey_Score', 'PerformanceGroup', 'EngagementGroup',
       'SatisfactionGroup', 'Work-Life_BalanceGroup', 'Total_Survey_Grou
p',
       'Age', 'JobFunction_LE', 'PositionTitle_LE', 'TenureGroup_'],
      dtype='object')
```

In [402]:

```
features = ['PositionTitle_LE', 'Faculty', 'EmployeeType', 'JobFunctior
       ,'StaffType', 'EmploymentDuration_months','Engagement Score', 'S

```

In [403]:

```
features = list(dict.fromkeys(features))
categorical_features = [col for col in features if df[col].dtype == 'ob
numerical_features = [col for col in features if col not in categorical

```

In [404]:

```
numerical_features
```

Out[404]:

```
['PositionTitle_LE',
 'JobFunction_LE',
 'Current Employee Rating',
 'EmploymentDuration_months',
 'Engagement Score',
 'Satisfaction Score',
 'Work-Life Balance Score',
 'TenureGroup_',
 'Total_Survey_Score',
 'Age']
```

In [405]:

```
categorical_features
```

Out[405]:

```
['Faculty', 'EmployeeType', 'GenderCode', 'StaffType']
```

In [406]:

```
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse=False), c
        ('num', 'passthrough', numerical_features)
    ]
)
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', XGBClassifier(use_label_encoder=False, eval_metric='
])
```

In [407]:

```
X_train, X_test, y_train, y_test = train_test_split(
    df[features], df['TerminatedStaff'], test_size=0.2, random_state=42

```

In [408]:

```
import joblib
pipeline.fit(X_train, y_train)
joblib.dump(pipeline, './xgb_pipeline_model.pkl')
#pipeline = joblib.load('xgb_pipeline_model.pkl')
y_pred = pipeline.predict(X_test)
```

```
C:\Users\omas0005\AppData\Local\anaconda3\Lib\site-packages\sklearn\prepro
cessing\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_o
utput` in version 1.2 and will be removed in 1.4. `sparse_output` is ignor
ed unless you leave `sparse` to its default value.
  warnings.warn(
C:\Users\omas0005\AppData\Local\anaconda3\Lib\site-packages\xgboost\traini
ng.py:183: UserWarning: [10:56:48] WARNING: C:\actions-runner\_work\xgboos
t\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

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

In [409]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       293
           1       0.86      0.82      0.84       307

    accuracy                           0.84       600
   macro avg       0.84      0.84      0.84       600
weighted avg       0.84      0.84      0.84       600
```

In [410]:

```
# Predict the probability
```

In [411]:

```
df_active = df[df['TerminatedStaff'] == 0]
```

In [412]:

```
X_active = df_active[features]
pred_leave = pipeline.predict(X_active)
prob_leave = pipeline.predict_proba(X_active)[:, 1]
df_active = df_active.copy()
df_active['Predicted_Terminated'] = pred_leave
df_active['Probability_Terminated'] = prob_leave
```

In [413]:

```
df_active
```

Out[413]:

	Unnamed: 0	FirstName	LastName	StartDate	ExitDate	PositionTitle	Supervisor	
0	0	Aaden	Mercer	2023-07-26	NaN	Senior Research Fellow	Victoria Jacobs	aaden
1	1	Aaliyah	Watts	2020-05-09	NaN	Senior Lecturer	Jared Whitehead	aaliya
2	2	Aarav	Espinoza	2019-11-11	NaN	Officer	David Ali	aarav.e
3	3	Aaron	Tapia	2023-01-20	NaN	Officer	Michael Reed	aar
6	6	Abagail	Moran	2019-08-24	NaN	Officer	Caitlin Stokes	abagai
...	...	...	...	...	...	...	...	
2987	2987	Zariah	Black	2019-11-13	NaN	Officer	Jenna Weaver	zaria
2991	2991	Zayne	Hunter	2019-11-11	NaN	Professor	Lisa Jordan	zayne
2996	2996	Zoey	Spence	2020-05-06	NaN	Senior Lecturer	Ian Price	zoey
2997	2997	Zoie	Logan	2019-04-19	NaN	Lecturer	Heidi Terry	zo
2998	2998	Zoie	Mercado	2020-12-01	NaN	Lecturer	David Young	zoie.n

1467 rows × 48 columns

In [414]:

```
df_active.to_csv("./active_staff_prediction.csv",index=False)
```

In [415]:

```python
# Select features from the whole dataframe
X_all = df[features]

# Predict termination class on entire data
pred_all = pipeline.predict(X_all)

# Predict probability for termination (class 1)
prob_all = pipeline.predict_proba(X_all)[:, 1]

# Add predictions and probabilities back to the dataframe
df['Predicted_Terminated'] = pred_all
df['Probability_Terminated'] = prob_all

# Evaluate performance on full dataset
from sklearn.metrics import classification_report, accuracy_score

print(classification_report(df['TerminatedStaff'], df['Predicted_Termir



```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      1467
           1       0.97      0.96      0.97      1533

    accuracy                           0.97      3000
   macro avg       0.97      0.97      0.97      3000
weighted avg       0.97      0.97      0.97      3000
```

In [416]:
```
df
```
Out[416]:

	Unnamed: 0	FirstName	LastName	StartDate	ExitDate	PositionTitle	Supervisor	
0	0	Aaden	Mercer	2023-07-26	NaN	Senior Research Fellow	Victoria Jacobs	aaden
1	1	Aaliyah	Watts	2020-05-09	NaN	Senior Lecturer	Jared Whitehead	aaliya
2	2	Aarav	Espinoza	2019-11-11	NaN	Officer	David Ali	aarav.e
3	3	Aaron	Tapia	2023-01-20	NaN	Officer	Michael Reed	aar
4	4	Aaron	Weber	2020-04-06	2023-07-24	Lecturer	Melanie Garcia	aaro
...	...	...	...	...	...	...	...	
2995	2995	Zoey	Page	2019-05-06	2022-04-09	Professor	William Newman	zo
2996	2996	Zoey	Spence	2020-05-06	NaN	Senior Lecturer	Ian Price	zoey
2997	2997	Zoie	Logan	2019-04-19	NaN	Lecturer	Heidi Terry	zo
2998	2998	Zoie	Mercado	2020-12-01	NaN	Lecturer	David Young	zoie.n
2999	2999	Zoie	Rowland	2022-02-08	2022-07-22	Senior Lecturer	Mrs. Rachel Henry	zoie.

3000 rows × 48 columns

In [417]:
```
print(f"Accuracy on full data: {accuracy_score(df['TerminatedStaff'], c

```
```
Accuracy on full data: 0.9680
```

In [418]:
```
df.to_csv("./Predict_Terminations.csv",index=False)
```

In [419]:
```
print(df[ ['TerminatedStaff', 'Predicted_Terminated', 'Probability_Term
```
```
   TerminatedStaff  Predicted_Terminated  Probability_Terminated
0                0                     0                0.062233
1                0                     0                0.000448
2                0                     0                0.000091
3                0                     0                0.044958
4                1                     1                0.851542
```

In [420]:

```python
# 1. Get feature names after preprocessing
# For OneHotEncoder, get categories and build feature names
ohe = pipeline.named_steps['preprocessor'].named_transformers_['cat']
ohe_feature_names = ohe.get_feature_names_out(categorical_features)

# Combine with numerical feature names (passed through)
feature_names = list(ohe_feature_names) + numerical_features

# 2. Get the trained XGBClassifier model
model = pipeline.named_steps['classifier']

# 3. Get feature importance scores (gain or weight or cover)
importance_dict = model.get_booster().get_score(importance_type='gain')

# 4. Map feature names to importance scores (note: importance keys migh
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': [importance_dict.get(f'f{i}', 0) for i in range(len(f
})

# Sort by importance descending
importance_df = importance_df.sort_values(by='Importance', ascending=Fa

print(importance_df)

```

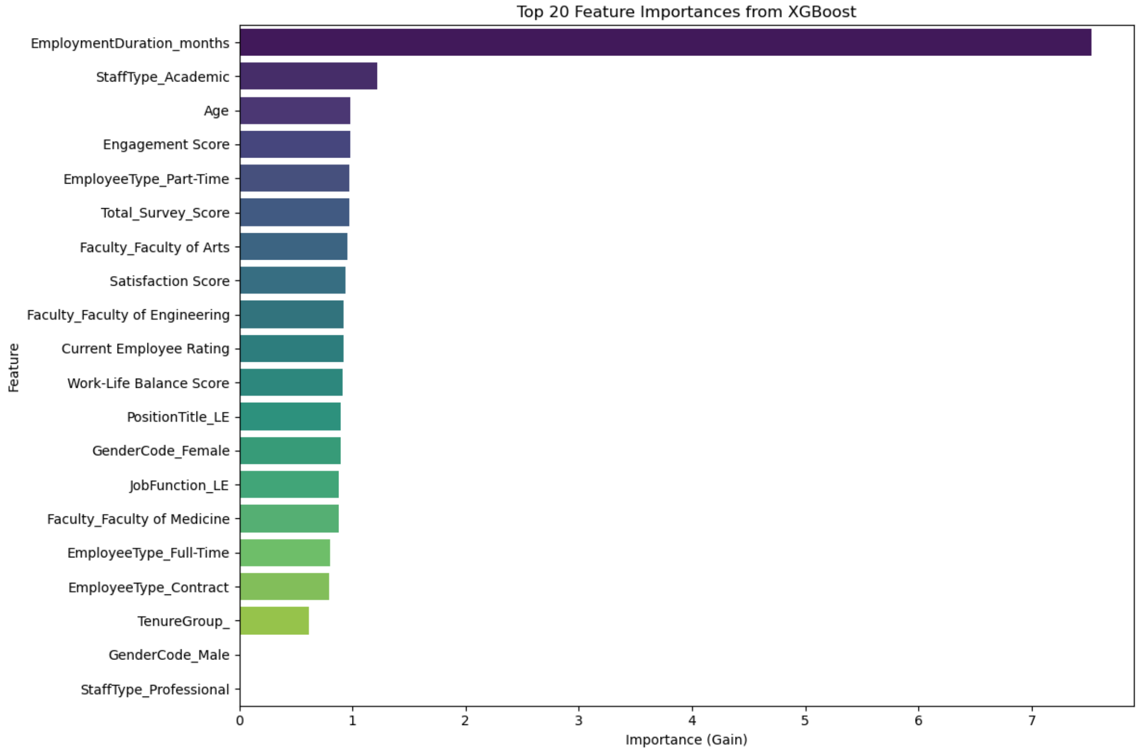
```
                       Feature  Importance
0      EmploymentDuration_months    7.525079
1             StaffType_Academic    1.220766
2                            Age    0.980411
3               Engagement Score    0.978649
4         EmployeeType_Part-Time    0.970561
5             Total_Survey_Score    0.970504
6        Faculty_Faculty of Arts    0.954568
7             Satisfaction Score    0.935931
8 Faculty_Faculty of Engineering    0.924429
9        Current Employee Rating    0.922855
10       Work-Life Balance Score    0.909402
11              PositionTitle_LE    0.900780
12             GenderCode_Female    0.894735
13                JobFunction_LE    0.881341
14   Faculty_Faculty of Medicine    0.876819
15        EmployeeType_Full-Time    0.799365
16         EmployeeType_Contract    0.794673
17                  TenureGroup_    0.613771
18               GenderCode_Male    0.000000
19        StaffType_Professional    0.000000
```

In [421]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming importance_df is created as before
plt.figure(figsize=(12, 8))
sns.barplot(data=importance_df.head(20), x='Importance', y='Feature', p
plt.title('Top 20 Feature Importances from XGBoost')
plt.xlabel('Importance (Gain)')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

```


Top 20 Feature Importances from XGBoost
EmploymentDuration_months
StaffType_Academic
Age
Engagement Score
EmployeeType_Part-Time
Total_Survey_Score
Faculty_Faculty of Arts
Satisfaction Score
Faculty_Faculty of Engineering
Current Employee Rating
Work-Life Balance Score
PositionTitle_LE
GenderCode_Female
JobFunction_LE
Faculty_Faculty of Medicine
EmployeeType_Full-Time
EmployeeType_Contract
TenureGroup_
GenderCode_Male
StaffType_Professional
Feature
Importance (Gain)