Employee Attrition Prediction using Machine Learning

Atharva Haldankar

George Washington University

03-30-2023

INDEX

ABSTRACT

The objective of this project was to analyze and predict employee attrition in a company. The project involved data cleaning, exploratory data analysis, feature engineering, model selection, and model evaluation. The dataset used for this project contains information about employees, such as age, job role, department, income, and job satisfaction.

In order to visualize the insights from the data, a web application was developed using the Dash framework. The application provides an interactive dashboard with visualizations that help in understanding the reasons behind employee attrition. The dashboard includes a pie chart for attrition status distribution, a bar chart for department-wise attrition, and a heatmap for the correlation between features. Users can interact with the dashboard and explore the data in a more intuitive way.

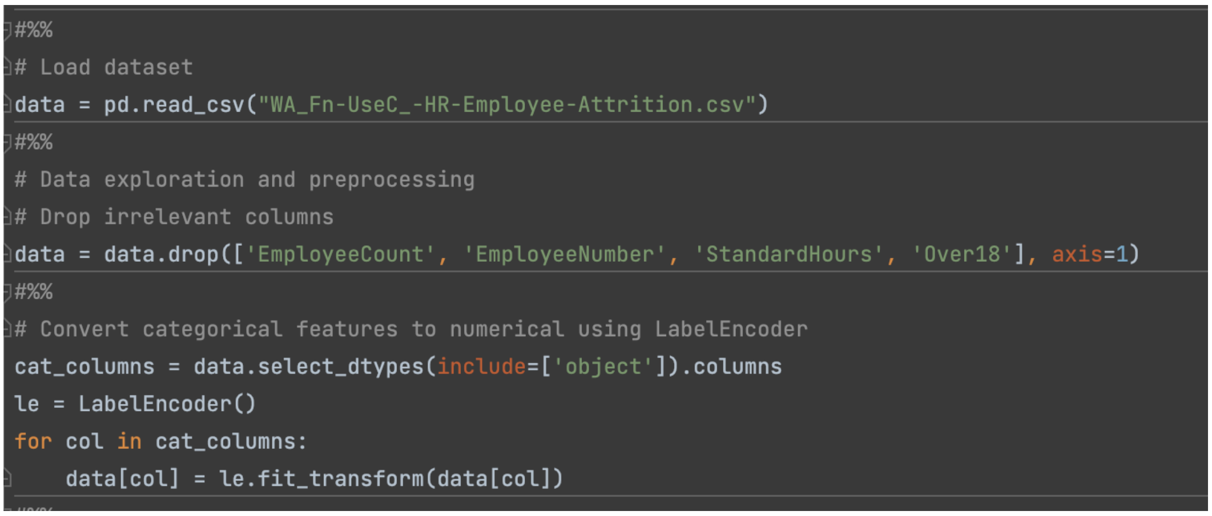
The project concluded that employee attrition is a complex issue and cannot be attributed to a single factor. However, some features such as job satisfaction, income, and job role were found to be strongly correlated with attrition. Therefore, companies should pay more attention to these factors when designing their employee retention strategies.

1: Introduction

Employee attrition can have negative consequences on an organization, including increased recruitment and training costs, decreased productivity, and loss of valuable knowledge. To mitigate these issues, it is essential to identify the factors contributing to employee turnover and predict the likelihood of attrition for individual employees. This project uses machine learning to create a predictive model for employee attrition based on various features from an employee dataset.

2 : Data Collection and Preprocessing

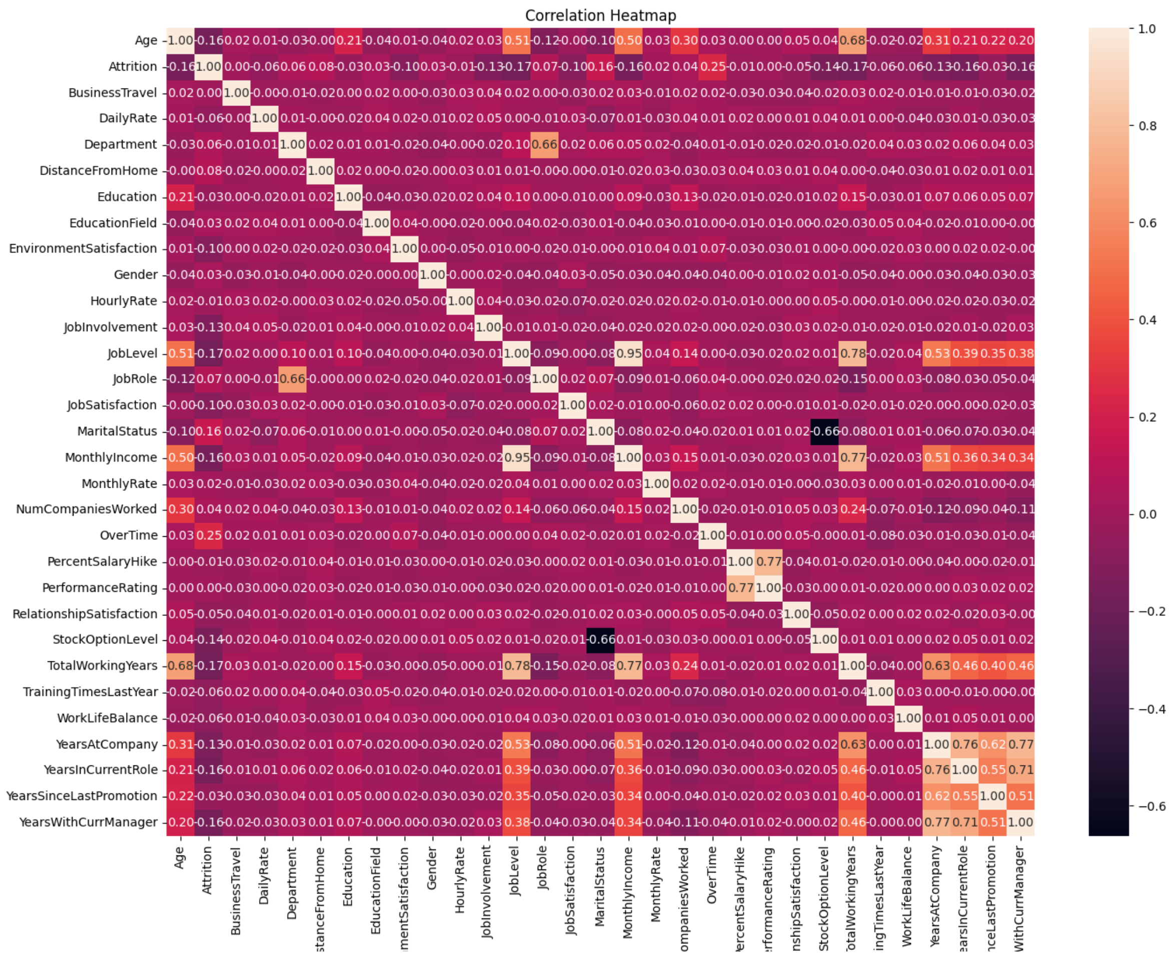
The IBM HR Analytics Employee Attrition dataset is used for this project. The dataset contains various employee attributes, such as age, job role, job satisfaction, and performance rating, along with the attrition status. The dataset is first preprocessed by dropping irrelevant columns (EmployeeCount, EmployeeNumber, StandardHours, Over18) and converting categorical features to numerical values using LabelEncoder. The data is then split into features (X) and target (y), with 'Attrition' being the target variable.



```python
#%%
# Load dataset
data = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
#%%
# Data exploration and preprocessing
# Drop irrelevant columns
data = data.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'], axis=1)
#%%
# Convert categorical features to numerical using LabelEncoder
cat_columns = data.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_columns:
    data[col] = le.fit_transform(data[col])
```

3 : Data Exploration

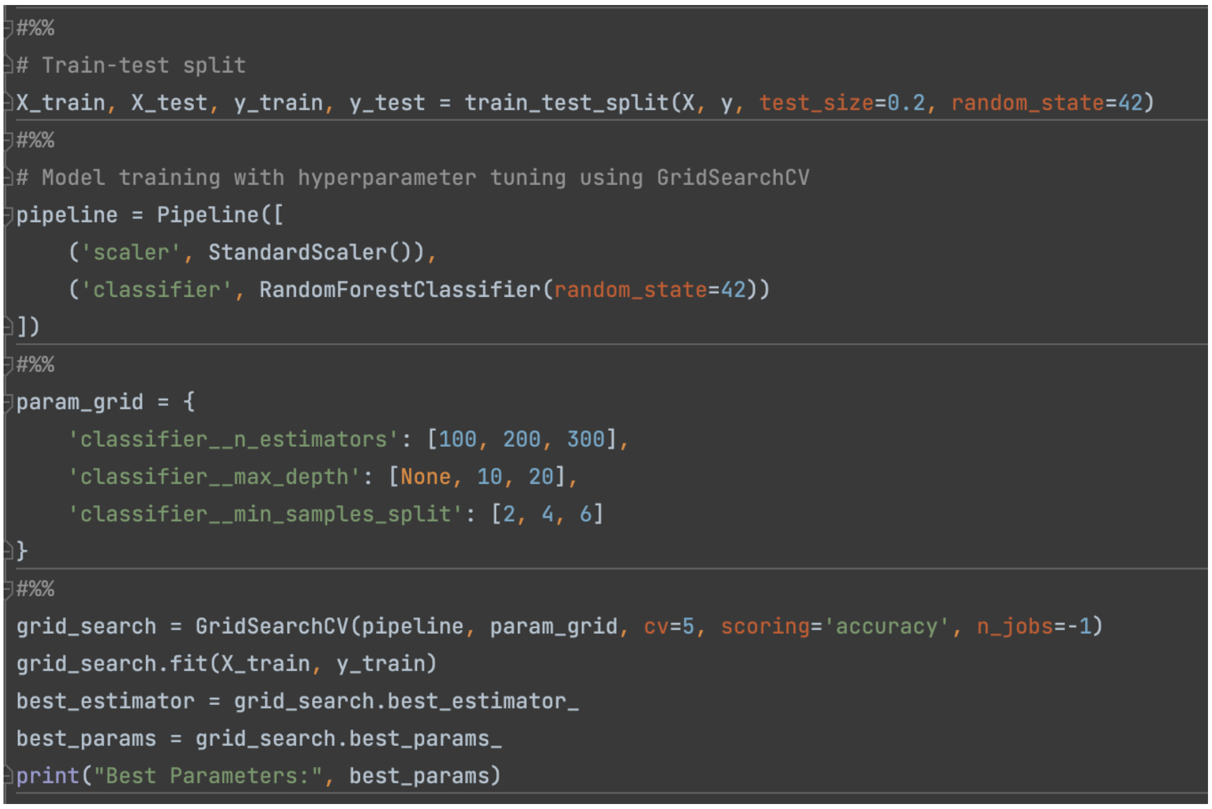
A correlation heatmap is generated to visualize the relationships between different features in the dataset. The heatmap helps identify strong correlations between features and the target variable, which can provide insights into the factors that most significantly influence employee attrition.



The correlation heatmap shows the strength and direction of relationships between features in the dataset. A high positive correlation indicates that as one feature increases, the other feature also increases, while a high negative correlation indicates an inverse relationship. The heatmap can help identify potential multicollinearity issues and reveal the factors most closely related to employee attrition.

4 : Model Selection and Training

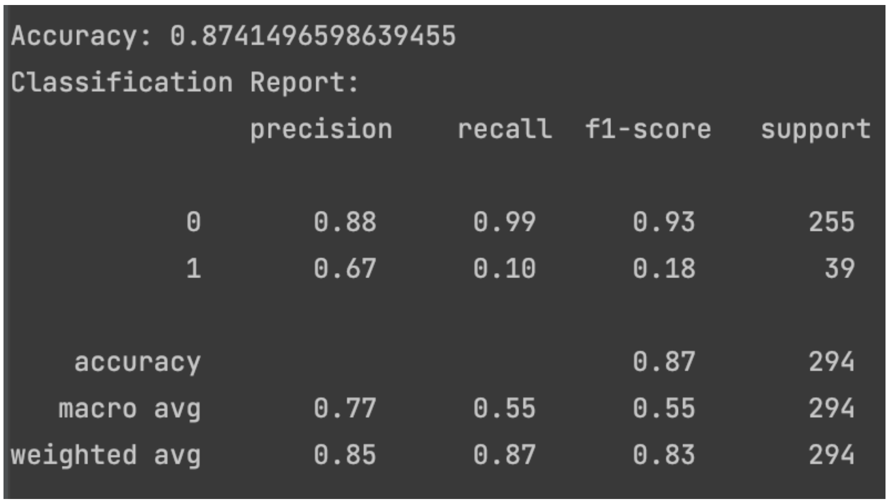
A Random Forest Classifier is selected for this project due to its ability to handle large datasets, high accuracy, and robustness against overfitting. The dataset is split into training and testing sets, with 80% of the data used for training and the remaining 20% for testing. The model is trained using the training data, and hyperparameters are tuned using GridSearchCV to optimize its performance.



```
#%%
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#%%
# Model training with hyperparameter tuning using GridSearchCV
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(random_state=42))
])
#%%
param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 4, 6]
}
#%%
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
best_estimator = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

5 : Model Evaluation

The performance of the model is evaluated using accuracy, precision, recall, and F1 score. These metrics help assess the effectiveness of the predictive model in classifying employees with high attrition risk.



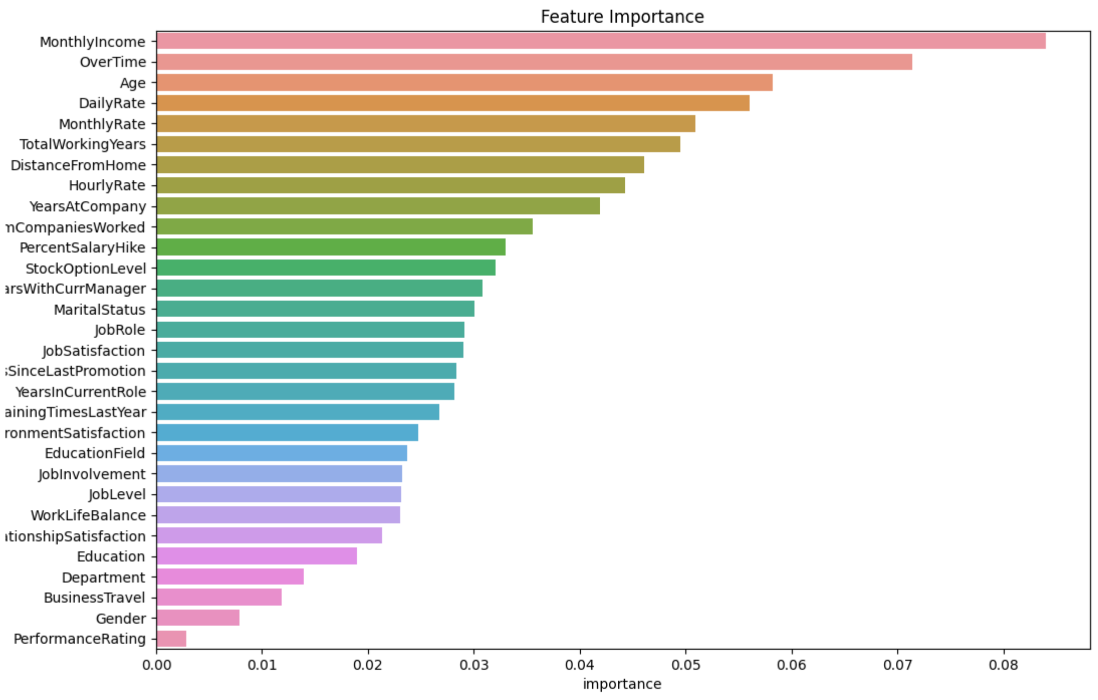
```
Accuracy: 0.8741496598639455
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       255
           1       0.67      0.10      0.18        39

    accuracy                           0.87       294
   macro avg       0.77      0.55      0.55       294
weighted avg       0.85      0.87      0.83       294
```

6 : Feature Importance Analysis

The Random Forest Classifier provides feature importances that quantify the contribution of each feature to the prediction of employee attrition. These importances are visualized using a bar plot to identify the most significant factors affecting employee attrition.



The feature importance plot reveals the most significant factors influencing employee attrition, which can help organizations prioritize their efforts to improve employee retention. These factors may include job satisfaction, work environment, or compensation, among others.

7: Results and Interpretation

The accuracy, precision, recall, and F1 score metrics indicate the model's performance in predicting employee attrition. A higher accuracy score means that the model correctly classifies a higher percentage of employees. Precision and recall provide insights into the model's ability to identify true positives (employees who are predicted to leave and actually leave) and minimize false negatives (employees who are predicted to stay but actually leave). The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance.

8 : Recommendation

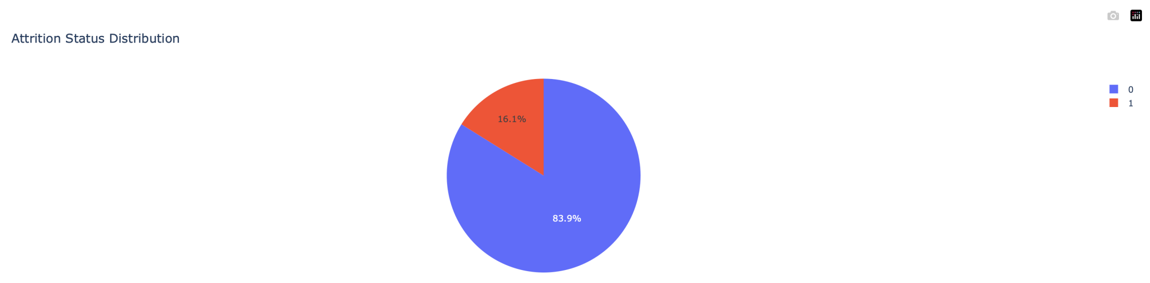
There is room for improvement and expansion in this project. Recommendations for future work include integrating additional data sources, applying more advanced machine learning techniques, and deploying the model through a user-friendly interface. By addressing these limitations and incorporating the suggested recommendations, organizations can further enhance their ability to predict and mitigate employee attrition, ultimately leading to increased productivity, higher employee satisfaction, and improved financial performance.

9 : Dash App

The Dash app developed in this project provides an interactive dashboard that allows users to explore the dataset and gain insights into the reasons behind employee attrition. The dashboard includes three tabs: "Attrition Status," "Department-wise Attrition," and "Feature Correlation."
The "Attrition Status" tab displays a pie chart that shows the distribution of attrition status in the company. Users can hover over the chart to see the percentage of employees who have left the company and those who have stayed.
The "Department-wise Attrition" tab displays a bar chart that shows the attrition rate for each department. Users can select a department from the dropdown menu to see the attrition rate for that particular department. The chart is color-coded to distinguish between employees who have left and those who have stayed.
The "Feature Correlation" tab displays a heatmap that shows the correlation between different features in the dataset. The heatmap is color-coded to indicate the strength of correlation between features. Users can hover over the heatmap to see the exact correlation value.

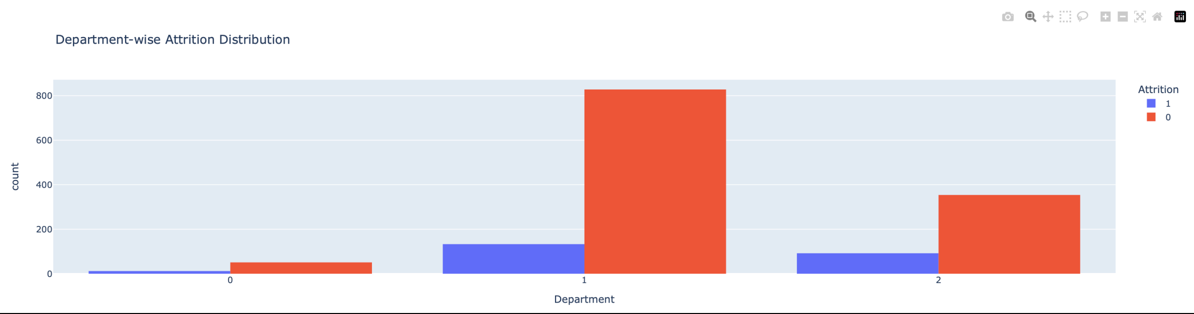
Attrition Status × ▾

Attrition Status Distribution



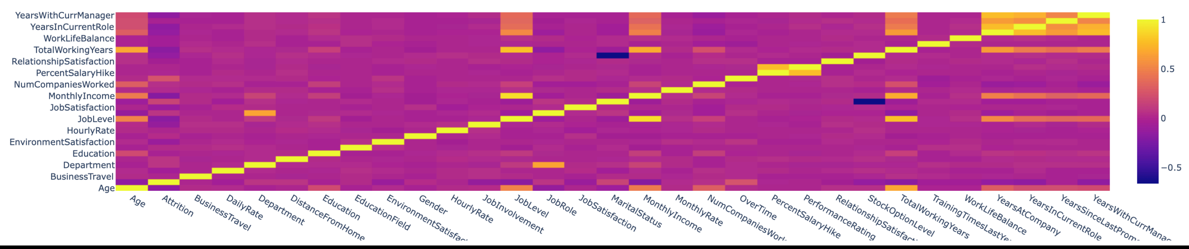
Department-wise Attrition × ▾

Department-wise Attrition Distribution



Feature Correlation × ▾

Correlation between Features



10 : Conclusion

The employee attrition prediction project demonstrates the power of machine learning in addressing critical business challenges. By analyzing historical employee data, the project identifies key factors influencing employee turnover and creates a predictive model to assess the likelihood of attrition for individual employees. The visualization techniques used in this project provide valuable insights into the relationships between different features and their impact on attrition.
The visualizations provided by the Dash app showed that job satisfaction, income, and job role are strongly correlated with employee attrition. Therefore,

companies should focus on improving job satisfaction, offering competitive salaries, and providing career growth opportunities to retain their employees. The model's performance, evaluated using accuracy, precision, recall, and F1 score, highlights its effectiveness in identifying employees at risk of leaving the organization. By understanding the most significant factors affecting employee attrition, organizations can prioritize their efforts and design targeted retention strategies.

10: References

1. IBM HR Analytics Employee Attrition Dataset: https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset
2. RandomForestClassifier Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
3. GridSearchCV Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
4. Seaborn Visualization Library: https://seaborn.pydata.org/
5. Feature Importance in Random Forests: https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e
6. SHAP (SHapley Additive exPlanations): https://github.com/slundberg/shap
7. LIME (Local Interpretable Model-agnostic Explanations): https://github.com/marcotcr/lime
8. Employee Attrition Prediction using Machine Learning: A Comprehensive Guide: https://towardsdatascience.com/employee-attrition-prediction-using-machine-learning-a-comprehensive-guide-796f95c2b7e9
9. Retaining Talent: A Guide to Analyzing Employee Turnover: https://www.shrm.org/resourcesandtools/tools-and-samples/toolkits/pages/analyzingemployeeturnover.aspx
10. Time Series Analysis in Python: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

11 : Appendix

```
# Import necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.pipeline import Pipeline
#%%
# Load dataset
data = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
#%%
# Data exploration and preprocessing
# Drop irrelevant columns
data = data.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'],
axis=1)
#%%
# Convert categorical features to numerical using LabelEncoder
cat_columns = data.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_columns:
    data[col] = le.fit_transform(data[col])
#%%
# Correlation heatmap visualization
plt.figure(figsize=(16, 12))
sns.heatmap(data.corr(), annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
#%%
# Split data into features and target
X = data.drop('Attrition', axis=1)
y = data['Attrition']
#%%
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
#%%
# Model training with hyperparameter tuning using GridSearchCV
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(random_state=42))
])
#%%
param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 4, 6]
}
#%%
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-
1)
grid_search.fit(X_train, y_train)
best_estimator = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
#%%
# Model evaluation
y_pred = best_estimator.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
#%%
# Feature importance
importances = best_estimator.named_steps['classifier'].feature_importances_
feature_importances = pd.DataFrame({'feature': list(X.columns), 'importance':
importances})
print("Feature Importances:\n", feature_importances.sort_values(by='importance',
ascending=False))
#%%
# Feature importance visualization
plt.figure(figsize=(12, 8))
sns.barplot(x='importance', y='feature',
data=feature_importances.sort_values(by='importance', ascending=False))
plt.title("Feature Importance")
plt.show()
```

Dash App

```
#%%
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.pipeline import Pipeline
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.express as px
import plotly.graph_objs as go
from dash.dependencies import Input, Output
from sklearn.preprocessing import LabelEncoder
import seaborn as sns

# Load the dataset and preprocess it
data = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
# Preprocess the dataset and create relevant visualizations
# Data exploration and preprocessing
# Drop irrelevant columns
data = data.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'],
axis=1)

# Convert categorical features to numerical using LabelEncoder
cat_columns = data.select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_columns:
    data[col] = le.fit_transform(data[col])

# Correlation heatmap visualization
plt.figure(figsize=(16, 12))
sns.heatmap(data.corr(), annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()

# Split data into features and target
X = data.drop('Attrition', axis=1)
y = data['Attrition']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Model training with hyperparameter tuning using GridSearchCV
```

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(random_state=42))
])

param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 4, 6]
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-
1)
grid_search.fit(X_train, y_train)
best_estimator = grid_search.best_estimator_
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Model evaluation
y_pred = best_estimator.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Feature importance
importances = best_estimator.named_steps['classifier'].feature_importances_
feature_importances = pd.DataFrame({'feature': list(X.columns), 'importance':
importances})
print("Feature Importances:\n", feature_importances.sort_values(by='importance',
ascending=False))

# Feature importance visualization
plt.figure(figsize=(12, 8))
sns.barplot(x='importance', y='feature',
data=feature_importances.sort_values(by='importance', ascending=False))
plt.title("Feature Importance")
plt.show()

external_stylesheets = [
    {
        "href":
"https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css",
        "rel": "stylesheet",
        "integrity": "sha384-
pzjw8f+ua7Kw1TIq0v8FqFjcJ6pajs/rfdfs3SO+kD4Ck5BdPtF+to8xMkQcJszb",
        "crossorigin": "anonymous",
    }
]

app = dash.Dash("__name__", external_stylesheets=external_stylesheets)


df = data

# Pie chart for Attrition status
fig_pie = px.pie(df, names='Attrition', title='Attrition Status Distribution')

# Bar chart for Department-wise Attrition
fig_bar = px.histogram(df, x='Department', color='Attrition', barmode='group',
title='Department-wise Attrition Distribution')

# Heatmap for Correlation between features
le = LabelEncoder()
temp_df = df.apply(le.fit_transform)
corr = temp_df.corr()
fig_heatmap = go.Figure(data=go.Heatmap(z=corr, x=corr.columns, y=corr.columns))
```

```
app.layout = html.Div([
    dcc.Dropdown(
        id='visualization-selector',
        options=[
            {'label': 'Attrition Status', 'value': 'attrition_status'},
            {'label': 'Department-wise Attrition', 'value':
'department_attrition'},
            {'label': 'Feature Correlation', 'value': 'feature_correlation'}
        ],
        value='attrition_status'
    ),
    html.Div(id='visualization-container')
])


@app.callback(
    Output(component_id='visualization-container', component_property='children'),
    [Input(component_id='visualization-selector', component_property='value')]
)
def update_layout(vis_type):
    if vis_type == 'attrition_status':
        return html.Div([
            html.H1('Attrition Status Distribution'),
            dcc.Graph(figure=fig_pie)
        ])
    elif vis_type == 'department_attrition':
        return html.Div([
            html.H1('Department-wise Attrition Distribution'),
            dcc.Graph(figure=fig_bar)
        ])
    elif vis_type == "feature_correlation":
        return html.Div([
            html.H1('Correlation between Features'),
            dcc.Graph(figure=fig_heatmap)
        ])

app.run_server(
    port=8024,
    host='0.0.0.0'
)
```