Inspiration

We've included some open-ended questions that you can explore and try to address through creating Tableau visualizations, or R or Python analyses. Good luck and enjoy the learning!

- Is there any relationship between who a person works for and their performance score?
- What is the overall diversity profile of the organization?
- What are our best recruiting sources if we want to ensure a diverse organization?
- Can we predict who is going to terminate and who isn't? What level of accuracy can we achieve on this?
- Are there areas of the company where pay is not equitable?

There are so many other interesting questions that could be addressed through this interesting data set. Dr. Patalano and I look forward to seeing what we can come up with.

If you have any questions or comments about the dataset, please do not hesitate to reach out to me on LinkedIn: http://www.linkedin.com/in/RichHuebner

You can also reach me via email at: Richard.Huebner@go.cambridgecollege.edu

HOW WE CREATE. PROJECT GUIDE LINE BY USING ML

Below is a comprehensive guide for a Human Resources Machine Learning project. This project will involve building a machine learning model to predict employee turnover, also known as employee attrition, using Python. The code will use common libraries such as Pandas, Scikit-Learn, and Matplotlib for data processing, model building, and visualization.

Human Resources Machine Learning Project: Predicting Employee Turnover

Objective:

To predict whether an employee will leave the company (attrition) based on various features such as age, job satisfaction, salary, etc.

Step-by-Step Guide

1. Data Collection and Preparation

For this project, we'll use a sample dataset. You can use an HR analytics dataset from sources like Kaggle or any other dataset you have.

Sample Data:

```
EmployeeID,Age,Gender,Department,Position,YearsAtCompany,JobSatisfaction,Salary,
Attrition

1,30,Male,Sales,Manager,5,4,75000,No
2,28,Female,Marketing,Executive,3,3,65000,Yes
```

2. Load and Explore the Data

```
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('employees.csv')

# Display the first few rows of the dataset
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Summary statistics
print(df.describe())
```

3. Data Preprocessing

```
# Encode categorical variables
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
df['Attrition'] = df['Attrition'].map({'No': 0, 'Yes': 1})

# One-hot encoding for department and position
df = pd.get_dummies(df, columns=['Department', 'Position'], drop_first=True)

# Drop irrelevant columns
df = df.drop(columns=['EmployeeID', 'Name'])

# Separate features and target variable
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

4. Split the Data into Training and Testing Sets

```
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

5. Build and Train the Model

We'll use a Random Forest classifier for this project.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)
```

```
# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

6. Feature Importance

```
# Get feature importances
importances = model.feature_importances_
feature_names = X.columns
feature_importances = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feature_importances = feature_importances.sort_values(by='Importance', ascending=False)

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances)
plt.title('Feature Importances')
plt.show()
```

7. Model Deployment

In a real-world scenario, you would save the model and deploy it using a framework like Flask or Django for making predictions on new data.

```
import joblib
```

```
# Save the model
joblib.dump(model, 'employee_attrition_model.pkl')

# Load the model
loaded_model = joblib.load('employee_attrition_model.pkl')

# Make predictions with the loaded model
new_predictions = loaded_model.predict(X_test)
```

Summary

In this project, we built a machine learning model to predict employee attrition using a Random Forest classifier. We started by loading and exploring the dataset, followed by data preprocessing, model building, training, and evaluation. Finally, we analyzed feature importances and discussed the steps for model deployment.

This project can be further enhanced by:

- Tuning hyperparameters using GridSearchCV.
- Trying different machine learning algorithms.
- Incorporating additional features.
- Building a more sophisticated model evaluation process.

Feel free to expand on this foundation based on your specific requirements and data availability.

Sample report

Introduction to the HR Dataset - Version 14

Updated April, 2021 The HR Dataset was designed by Drs. Rich Huebner and Carla Patalano to accompany a case study designed for graduate HR students studying HR metrics, measurement, and analytics.

Data Dictionary

Feature	Description	DataType
Employee Name	Employee's full name	Text
EmpID	Employee ID is unique to each employee	Text
MarriedID	Is the person married (1 or 0 for yes or no)	Binary
MaritalStatusID	Marital status code that matches the text field MaritalDesc	Integer
EmpStatusID	Employment status code that matches text field EmploymentStatus	Integer
DeptID	Department ID code that matches the department the employee works in	Integer
PerfScoreID	Performance Score code that matches the employee's most recent performance score	Integer
FromDiversityJobFairID	Was the employee sourced from the Diversity job fair? 1 or 0 for yes or no	Binary
Salary	The person's yearly salary. $ U.S. Dollars	Float
Termd	Has this employee been terminated - 1 or 0	Binary
PositionID	An integer indicating the person's position	Integer
Position	The text name/title of the position the person has	Text
State	The state that the person lives in	Text
Zip	The zip code for the employee	Text
DOB	Date of Birth for the employee	Date
Sex	Sex - M or F	Text
MaritalDesc	The marital status of the person (divorced, single, widowed, separated, etc)	Text
CitizenDesc	Label for whether the person is a Citizen or Eligible NonCitizen	Text
HispanicLatino	Yes or No field for whether the employee is Hispanic/Latino	Text
RaceDesc	Description/text of the race the person identifies with	Text
DateofHire	Date the person was hired	Date

In [3]:
```
# Set display options to show all rows and columns
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
```

In [4]:
```
import warnings

# To ignore all warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
df.head(10)
```

Out[5]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversityJobFairID	Salary	Termd	PositionID	Position
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	0	62506	0	19	Production Technician I
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	0	104437	1	27	Sr. DBA
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	0	64955	1	20	Production Technician II

In [6]:

```
def get_df_info(df):
    print("\n\033[1mShape of DataFrame:\033[0m ", df.shape)
    print("\n\033[1mColumns in DataFrame:\033[0m ", df.columns.to_list())
    print("\n\033[1mData types of columns:\033[0m\n", df.dtypes)

    print("\n\033[1mInformation about DataFrame:\033[0m")
    df.info()

    print("\n\033[1mNumber of unique values in each column:\033[0m")
    for col in df.columns:
        print(f"\033[1m{col}\033[0m: {df[col].nunique()}")

    print("\n\033[1mNull values in columns:\033[0m")
    null_counts = df.isnull().sum()
    null_columns = null_counts[null_counts > 0]
    if len(null_columns) > 0:
        for col, count in null_columns.items():
            print(f"\033[1m{col}\033[0m: {count}")
    else:
        print("There are no null values in the DataFrame.")

    print("\n\033[1mNumber of duplicate rows:\033[0m ", df.duplicated().sum())
```

```
            print(f"\033[1m{col}\033[0m: {count}")
    else:
        print("There are no null values in the DataFrame.")

    print("\n\033[1mNumber of duplicate rows:\033[0m ", df.duplicated().sum())

    print("\n\033[1mDescriptive statistics of DataFrame:\033[0m\n",)
    return df.describe().transpose()

# Call the function
get_df_info(df)
```

Shape of DataFrame: (311, 36)

Columns in DataFrame: ['Employee_Name', 'EmpID', 'MarriedID', 'MaritalStatusID', 'GenderID', 'EmpStatusID', 'DeptID', 'PerfScoreID', 'FromDiversityJobFairID', 'Salary', 'Termd', 'PositionID', 'Position', 'State', 'Zip', 'DOB', 'Sex', 'MaritalDesc', 'CitizenDesc', 'HispanicLatino', 'RaceDesc', 'DateofHire', 'DateofTermination', 'TermReason', 'EmploymentStatus', 'Department', 'ManagerName', 'ManagerID', 'RecruitmentSource', 'PerformanceScore', 'EngagementSurvey', 'EmpSatisfaction', 'SpecialProjectsCount', 'LastPerformanceReview_Date', 'DaysLateLast30', 'Absences']

Data types of columns:

Employee_Name	object
EmpID	int64
MarriedID	int64

```
Data types of columns:
 Employee_Name                object
EmpID                         int64
MarriedID                     int64
MaritalStatusID               int64
GenderID                      int64
EmpStatusID                   int64
DeptID                        int64
PerfScoreID                   int64
FromDiversityJobFairID        int64
Salary                        int64
Termd                         int64
PositionID                    int64
Position                     object
State                        object
Zip                           int64
DOB                          object
Sex                          object
MaritalDesc                  object
CitizenDesc                  object
HispanicLatino               object
RaceDesc                     object
```

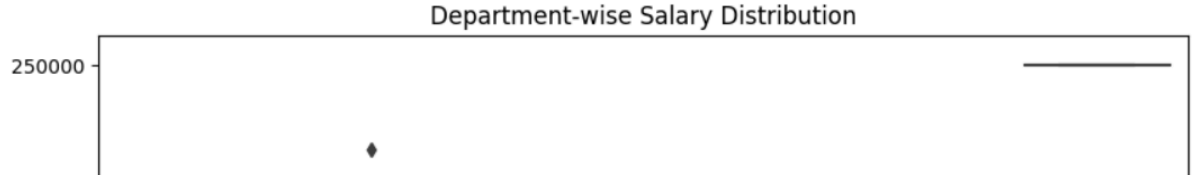
In [7]:

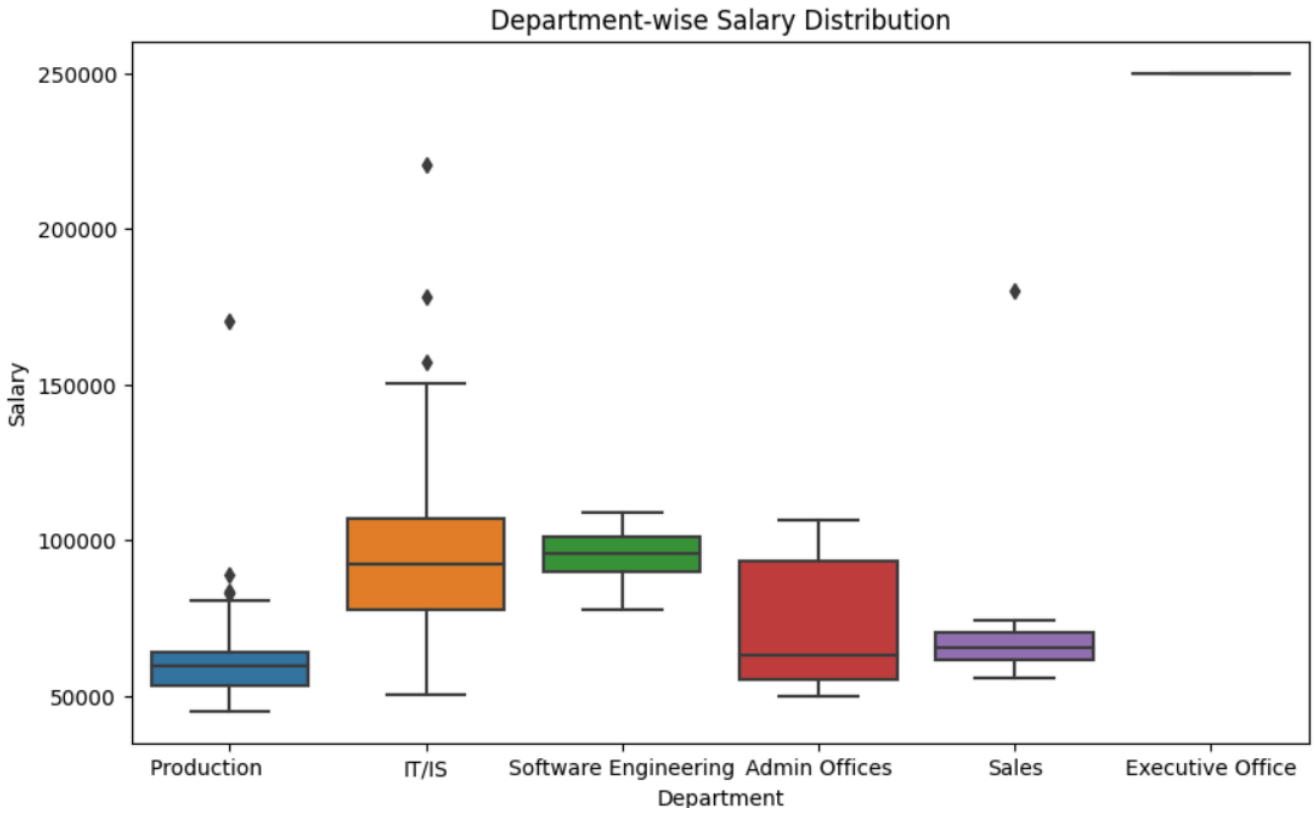
```
import seaborn as sns
import matplotlib.pyplot as plt
```

Anallysing the Salary

In [8]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(x="Department", y="Salary", data=df)
plt.title("Department-wise Salary Distribution")
plt.xlabel("Department")
plt.ylabel("Salary")
plt.show()
```




Department-wise Salary Distribution
250000
200000
150000
100000
50000
Salary
Production
IT/IS
Software Engineering
Admin Offices
Sales
Executive Office
Department