```
#code
from google.colab import files
uploaded = files.upload()
```

Choose Files HR_comma_sep.csv

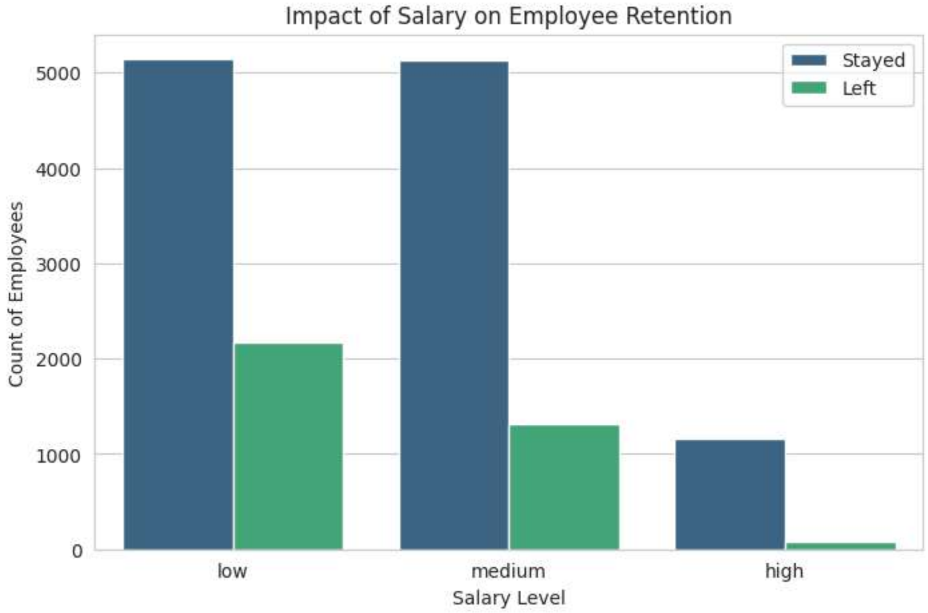
- **HR_comma_sep.csv**(text/csv) - 566785 bytes, last modified: 3/17/2025 - 100% done

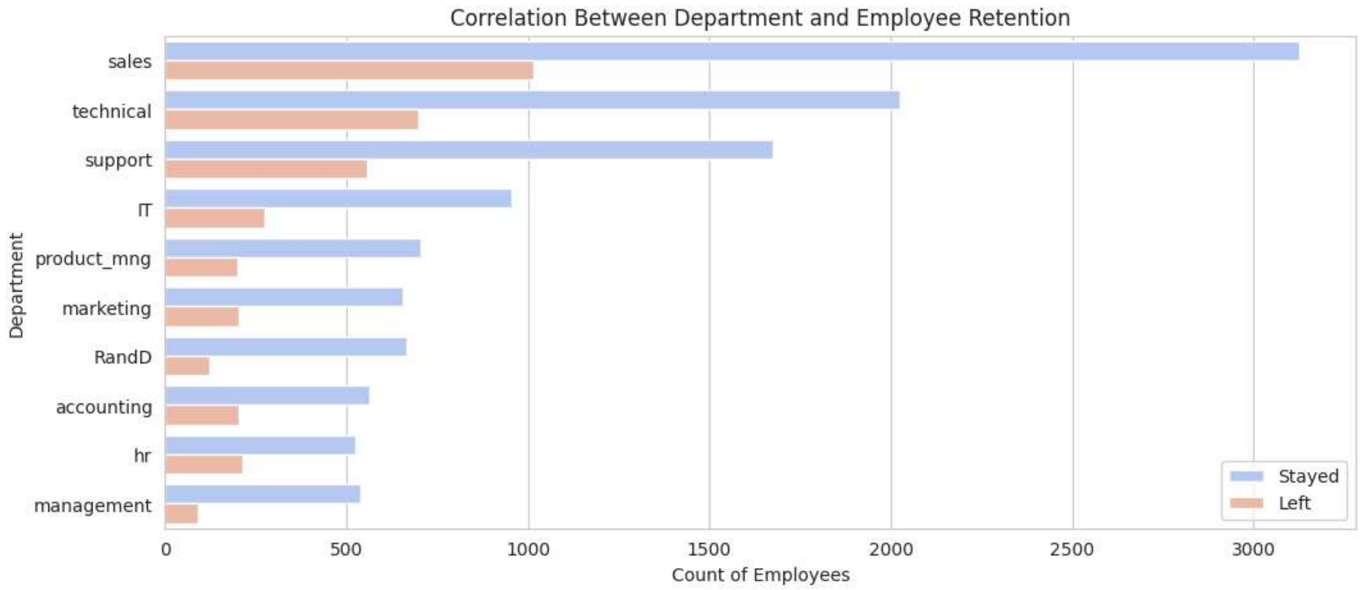
```
Saving HR_comma_sep.csv to HR_comma_sep.csv
```

```
#binary class
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("HR.csv")
print(df.head())

missing_values = df.isnull().sum()
# Display columns with missing values
print(missing_values[missing_values > 0])

# Set seaborn style
sns.set_style("whitegrid")

# Plot bar chart for salary vs retention
plt.figure(figsize=(8, 5))
sns.countplot(x="salary", hue="left", data=df, palette="viridis")
plt.xlabel("Salary Level")
plt.ylabel("Count of Employees")
plt.title("Impact of Salary on Employee Retention")
plt.legend(["Stayed", "Left"])
plt.show()

# Plot bar chart for department vs retention
plt.figure(figsize=(12, 5))
sns.countplot(y="Department", hue="left", data=df, palette="coolwarm", order=df["Department"].value_counts().index)
plt.xlabel("Count of Employees")
plt.ylabel("Department")
plt.title("Correlation Between Department and Employee Retention")
plt.legend(["Stayed", "Left"])
plt.show()

# Encode categorical variables
label_encoders = {}
for col in ["salary", "Department"]:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Select relevant features
features = ["satisfaction_level", "last_evaluation", "number_project", "average_montly_hours",
            "time_spend_company", "Work_accident", "promotion_last_5years", "salary", "Department"]
X = df[features]
y = df["left"]

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train logistic regression model
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

# Predict on test set
y_pred = log_reg.predict(X_test)

# Measure accuracy
accuracy = accuracy_score(y_test, y_pred)
```

```
    accuracy = accuracy_score(y_test, y_pred)
    classification_rep = classification_report(y_test, y_pred)

    # Print results
    print(f"Model Accuracy: {accuracy:.4f}")
    print("Classification Report:")
    print(classification_rep)
```

```
   satisfaction_level  last_evaluation  number_project  average_montly_hours  \
0                0.38             0.53               2                   157   
1                0.80             0.86               5                   262   
2                0.11             0.88               7                   272   
3                0.72             0.87               5                   223   
4                0.37             0.52               2                   159   

   time_spend_company  Work_accident  left  promotion_last_5years Department  \
0                   3              0     1                      0      sales   
1                   6              0     1                      0      sales   
2                   4              0     1                      0      sales   
3                   5              0     1                      0      sales   
4                   3              0     1                      0      sales   

   salary  
0     low  
1  medium  
2  medium  
3     low  
4     low  
Series([], dtype: int64)
```





```
Model Accuracy: 0.7577
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.92      0.85      2294
           1       0.47      0.23      0.31       706

    accuracy                           0.76      3000
   macro avg       0.63      0.57      0.58      3000
weighted avg       0.72      0.76      0.72      3000
```

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

# Load dataset
file_path = "HR.csv"  # Update this if needed
df = pd.read_csv(file_path)

# Display basic info
display(df.info())
display(df.head())

# Correlation heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap of Numerical Variables")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
```

```
#code2
from google.colab import files
uploaded = files.upload()
```

```
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
None
```

Choose Files zoo-data (1).csv

- **zoo-data (1).csv**(text/csv) - 4368 bytes, last modified: 3/17/2025 - 100% done

```
Saving zoo-data (1).csv to zoo-data (1).csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load the dataset
df_zoo = pd.read_csv("tr.csv")

# Drop 'animal_name' as it's not useful for classification
df_zoo = df_zoo.drop(columns=["animal_name"])

# Separate features and target variable
X = df_zoo.drop(columns=["class_type"])  # Features
y = df_zoo["class_type"]  # Target (class type)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Standardize numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train multinomial logistic regression model
model = LogisticRegression(multi_class="multinomial", max_iter=1000)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Measure accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.4f}")

# Print classification report
print("Classification Report:\n", classification_report(y_test, y_pred))

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=np.unique(y), yticklabels=np.unique(y))
plt.xlabel("Predicted Class")
plt.ylabel("Actual Class")
plt.title("Confusion Matrix for Zoo Dataset")
plt.show()
```