Project: Employee Attrition Prediction Pipeline

Business Problem

Employee attrition is a significant challenge for companies, leading to increased recruitment and training costs, loss of organizational knowledge, and decreased productivity. Predicting which employees are at risk of leaving allows HR departments to take proactive measures to retain valuable talent. This project aims to build an end-to-end ETL pipeline to predict employee attrition using various data engineering tools.

Project Overview

and possibly other tools like Kafka and Cassandra. The pipeline will extract data from various sources, transform it, load it into a data warehouse, and finally, analyze and predict employee attrition.

Data Sources

1. **HR Database**: Employee demographics, job role, department, salary, etc.
2. **Surveys**: Employee satisfaction, engagement scores, feedback, etc.
3. **Attendance Records**: Absenteeism, leaves taken, etc.
4. **Performance Metrics**: Appraisals, performance scores, etc.

Tasks Details

Step 1: Data Extraction

1. **Extract HR Data**: Use Airflow to schedule and extract employee data from a PostgreSQL database.
2. **Extract Survey Data**: Extract survey data stored in AWS S3 buckets.
3. **Extract Attendance Records**: Extract attendance data from a CSV file stored in AWS S3.
4. **Extract Performance Metrics**: Extract performance data from a Kafka stream.

Step 2: Data Transformation

1. **Data Cleaning**: Use Apache Spark to clean the data (handle missing values, correct data types, etc.).
2. **Data Integration**: Merge data from various sources using Spark.
3. **Feature Engineering**: Create new features that might be indicative of attrition (e.g., employee tenure, average performance score).

Step 3: Data Loading

1. **Load into Data Warehouse**: Load the cleaned and transformed data into a PostgreSQL or Cassandra database for further analysis.

Step 4: Data Analysis and Modeling

1. **Exploratory Data Analysis (EDA)**: Use SQL and Python (e.g., pandas, matplotlib) to understand the data and identify trends.
2. **Model Training**: Use Spark MLlib or another ML framework to train a predictive model for attrition.

Step 5: Model Deployment

1. **Deploy Model**: Deploy the trained model using a REST API (e.g., Flask) that HR systems can query to get attrition predictions.

Step 6: Monitoring and Maintenance

1. **Pipeline Monitoring**: Use Airflow's monitoring capabilities to ensure that the pipeline runs smoothly.

2. **Model Monitoring**: Track model performance over time and retrain as necessary.

Detailed Pipeline Implementation

Step 1: Data Extraction

Airflow DAG for Data Extraction

```python
from airflow import DAG
from airflow.operators.postgres_operator import PostgresOperator
from airflow.operators.python_operator import PythonOperator
from airflow.operators.s3_to_s3_operator import S3ToS3Operator
from airflow.operators.kafka_consumer_operator import KafkaConsumerOperator
from datetime import datetime

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'retries': 1,
}

dag = DAG('data_extraction', default_args=default_args, schedule_interval='@da

# Extract HR data
extract_hr_data = PostgresOperator(
    task_id='extract_hr_data',
    sql='SELECT * FROM hr_data',
    postgres_conn_id='hr_postgres',
    database='hr_db',
    dag=dag
)

# Extract Survey data from S3
extract_survey_data = S3ToS3Operator(
    task_id='extract_survey_data',
    source_s3_key='s3://source-bucket/surveys/',
    dest_s3_key='s3://destination-bucket/surveys/',
    aws_conn_id='aws_default',
    dag=dag
)

# Extract Attendance data from S3
extract_attendance_data = S3ToS3Operator(
    task_id='extract_attendance_data',
    source_s3_key='s3://source-bucket/attendance.csv',
    dest_s3_key='s3://destination-bucket/attendance.csv',
    aws_conn_id='aws_default',
    dag=dag
)

# Extract Performance data from Kafka
extract_performance_data = KafkaConsumerOperator(
    task_id='extract_performance_data',
    kafka_conn_id='kafka_default',
    topic='performance_data',
    group_id='performance_group',
    dag=dag
)
```

```
extract_hr_data >> extract_survey_data >> extract_attendance_data >> extract_p
```

Step 2: Data Transformation

Spark Job for Data Cleaning and Transformation

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

spark = SparkSession.builder \
    .appName("HR Data Transformation") \
    .getOrCreate()

# Load data
hr_data = spark.read.csv("s3://destination-bucket/hr_data.csv", header=True, i
survey_data = spark.read.csv("s3://destination-bucket/surveys/survey_data.csv'
attendance_data = spark.read.csv("s3://destination-bucket/attendance.csv", hea
performance_data = spark.read.json("path_to_kafka_data")

# Data Cleaning
hr_data = hr_data.na.drop()
survey_data = survey_data.na.fill({"satisfaction_score": 0})
attendance_data = attendance_data.na.fill({"absent_days": 0})

# Data Integration
data = hr_data.join(survey_data, "employee_id").join(attendance_data, "employe

# Feature Engineering
data = data.withColumn("tenure_years", col("tenure") / 12)
data = data.withColumn("high_performer", when(col("performance_score") > 85, 1

# Save transformed data
data.write.format("parquet").save("s3://destination-bucket/transformed_data/")
```

Step 3: Data Loading

Loading Data into PostgreSQL

```
from sqlalchemy import create_engine
import pandas as pd

engine = create_engine('postgresql://username:password@hostname/dbname')
transformed_data = pd.read_parquet("s3://destination-bucket/transformed_data/'

transformed_data.to_sql('employee_data', engine, if_exists='replace', index=Fa
```

Step 4: Data Analysis and Modeling

Training a Predictive Model

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Load transformed data
data = spark.read.parquet("s3://destination-bucket/transformed_data/")

# Feature columns
feature_cols = ["tenure_years", "satisfaction_score", "absent_days", "high_pe
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")

# Prepare data for training
data = assembler.transform(data)
train_data, test_data = data.randomSplit([0.7, 0.3], seed=42)

# Train model
rf = RandomForestClassifier(labelCol="attrition", featuresCol="features")
model = rf.fit(train_data)

# Evaluate model
predictions = model.transform(test_data)
evaluator = BinaryClassificationEvaluator(labelCol="attrition")
accuracy = evaluator.evaluate(predictions)

print(f"Model Accuracy: {accuracy}")
```

Step 5: Model Deployment

Deploy Model using Flask

```
from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)

# Load trained model
model = joblib.load("path_to_saved_model.pkl")

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([data['features']])
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Step 6: Monitoring and Maintenance

Airflow Monitoring

- Set up alerts for task failures.
- Use Airflow’s UI to monitor DAG runs.

Model Monitoring

- Track prediction accuracy over time.
- Set up a process to retrain the model periodically or when performance drops.

Conclusion

This end-to-end ETL pipeline allows HR departments to predict employee attrition effectively, enabling proactive measures to retain talent. By leveraging various data engineering tools like Airflow, Spark, and AWS S3, we ensure a robust and scalable solution.

Previous
ETL Pipline on HR Attrition Analysis

Next
ETL DAG's

Last updated 3 months ago