Une image contenant texte, Police, capture d’écran, carte de visite

Description générée automatiquement

# Academic Project Report

Machine Learning Pipeline in PySpark

Realized By : **Aziz Ayadi**

**Louai Azzouni**

**Rayen Ben Fathallah**

Supervised By : **Manel Zekri**

Academic Year : 2024 – 2025

**Objective**

The primary objective of this project is to build a machine learning model in PySpark to predict whether an individual's income exceeds $50,000 per year based on census data. This guide uses the "Census Income" dataset, incorporating data preprocessing, model building, evaluation, and hyperparameter tuning.

**Dataset Overview**

The dataset, referred to as "adult.csv," includes features such as age, work class, education level, occupation, marital status, and capital gain. The target variable, income, is a binary classification label that indicates whether a person's income is above or below $50,000.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

**Steps**

**1. Initialize SparkContext and SQLContext**

* **SparkContext (sc)**: Provides the core functionality for Spark jobs. In PySpark, SparkContext is required to establish the initial connection with Spark.
* **SQLContext (sqlContext)**: Enables Spark to interface with SQL-based data sources. It’s initialized to read the dataset and perform SQL-like transformations.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

**2. Load and Inspect Dataset**

* **Load Dataset**: The CSV file is read into a DataFrame using sqlContext.read.csv(), with header=True and inferSchema=True to automatically infer column types.
* **Schema Inspection**: The .printSchema() function provides an overview of data types and nullability, showing features such as age, workclass, fnlwgt, education, marital-status, and income.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

3. **Data Preprocessing**

* **Column Renaming**: Renaming columns for simplicity and consistency in naming conventions.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* **Type Conversion**: Conversion of continuous features to FloatType using a custom function. Important continuous features include age, fnlwgt, capital\_gain, education\_num, capital\_loss, and hours\_week.

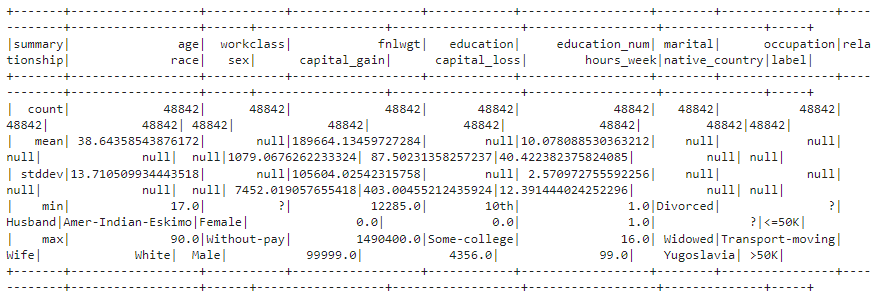
Une image contenant texte, capture d’écran, Police

Description générée automatiquement

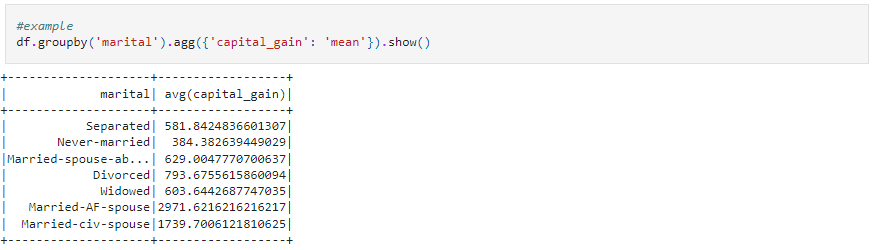
Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

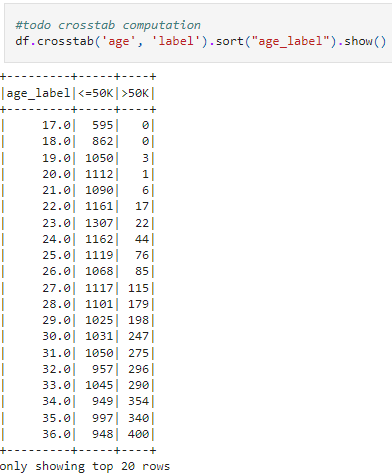
* **Descriptive Statistics**: Use .describe() to generate basic statistical summaries such as mean, standard deviation, min, and max for each column.



* **Grouping and Aggregation**: For example, computing the average capital\_gain by marital status provides insights into capital gains across different marital groups.



* **Crosstab Analysis**: Creates a cross-tabulation between age and income (<=50K or >50K), providing a count of individuals in each age group by income category.



* **Null Handling and Row Exclusion**: Removes rows with missing values using .dropna(). For example, records from native\_country with only one entry (like "Holand-Netherlands") are excluded to avoid issues in cross-validation.





Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

**4. Feature Engineering**

* **Age Transformation**: Create a new feature, age\_square, to capture potential non-linear relationships between age and income.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

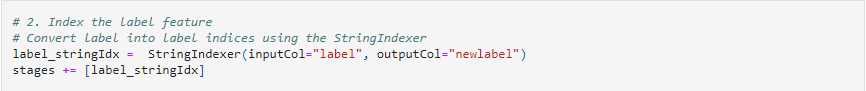
**5. Building the Data Processing Pipeline**

* **Categorical Feature Encoding**: Use StringIndexer and OneHotEncoderEstimator to encode categorical features, including workclass, education, marital, occupation, relationship, race, sex, and native\_country.

Une image contenant texte, capture d’écran, Police, Page web

Description générée automatiquement

* **Label Encoding**: The target variable income is converted into a binary label (label) using StringIndexer.

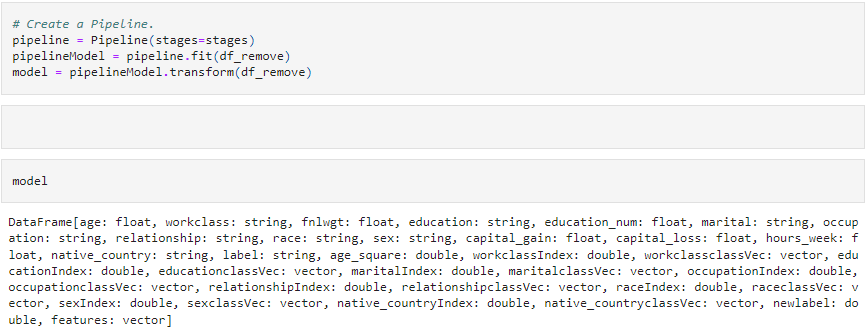


* **Feature Vectorization**: Use VectorAssembler to combine all transformed categorical and continuous features into a single vector, features.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

* **Pipeline Creation**: Instantiate a Pipeline that will execute all the stages sequentially.



**6. Model Building - Logistic Regression**

* **Dataset Preparation**: Split the dataset into an 80/20 train-test split for model validation.



* **Logistic Regression Initialization**: Initialize Logistic Regression with labelCol="label" and featuresCol="features".

Une image contenant texte, capture d’écran, document, Police

Description générée automatiquement

**7. Model Evaluation**

* **Accuracy Calculation**: Compute accuracy by comparing predicted labels to true labels in the test set.
* **ROC and Area Under ROC**: Evaluate the model’s performance using the ROC curve, where an area under the ROC of 0.89 indicates strong model performance.

Une image contenant texte, capture d’écran, Police

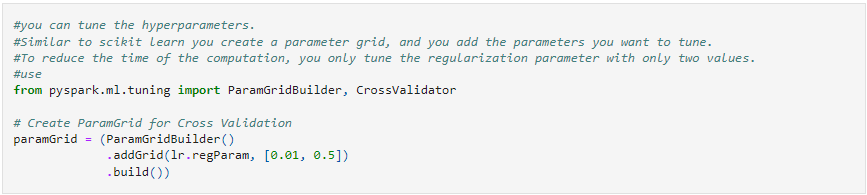
Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**8. Hyperparameter Tuning with Cross-Validation**

* **Parameter Grid**: Construct a ParamGridBuilder to test two values for the regParam parameter.

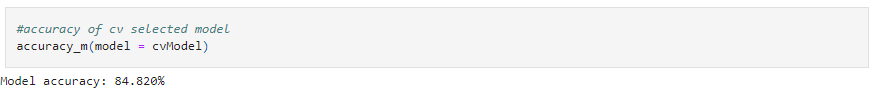


* **CrossValidator**: Set up a 5-fold cross-validation to identify the optimal regularization parameter and evaluate model accuracy.

Une image contenant texte, Police, Page web, capture d’écran

Description générée automatiquement

* **Best Model Accuracy**: The cross-validated model achieves an accuracy of 84.82%, demonstrating the effectiveness of tuning.



**Conclusion**

The completed PySpark ML pipeline demonstrates efficient handling of categorical data, preprocessing of features, and model evaluation in Spark. With a final accuracy of around 84.82% on the test set, this model effectively predicts income levels based on census data.

This pipeline can be expanded with additional feature engineering, model evaluation metrics, and alternative algorithms if needed. The entire process is scalable and easily adaptable for similar classification tasks in large datasets using Spark.