Artículo Científico: Análisis y Predicción de Precios de Laptops con Técnicas de Preprocesamiento, Regresión y Reducción de Dimensionalidad

Jorge Luis Chuquimia Parra Universidad Mayor de San Andres

Resumen

En este estudio se investigan las técnicas de preprocesamiento de datos, regresión con Random Forest, y reducción de dimensionalidad mediante PCA para predecir los precios de laptops en función de sus características. Se exploran técnicas de limpieza de datos, codificación de variables categóricas, normalización de datos, y la evaluación de un modelo de regresión, junto con la reducción de dimensionalidad para mejorar la precisión del modelo.

Summary

This study investigates data preprocessing, Random Forest regression, and dimensionality reduction using PCA to predict laptop prices based on their characteristics. Data cleaning techniques, categorical variable coding, data normalization, and evaluation of a regression model are explored, along with dimensionality reduction to improve model accuracy.

1. Introducción

La predicción de precios en productos tecnológicos como laptops puede ser una tarea desafiante debido a la variedad de características involucradas (marca, especificaciones técnicas, tipo de almacenamiento, etc.). Este artículo aborda el proceso de predicción del precio de las laptops utilizando un conjunto de datos disponible públicamente. En particular, se exploran técnicas de preprocesamiento de datos, balanceo de clases (cuando sea aplicable), y el uso de modelos de regresión y reducción de dimensionalidad.

2. Metodología

2.1. Preprocesamiento de Datos

Limpieza de Datos:
El primer paso del proceso fue la limpieza de datos, que incluyó la eliminación de valores faltantes y duplicados. Se optó por eliminar las filas con valores faltantes debido a la naturaleza de los datos y la importancia de tener un conjunto de entrenamiento limpio.

Codificación de Variables Categóricas:
Las variables categóricas del conjunto de datos, tales como 'Company', 'Product', 'TypeName', 'OS', entre otras, fueron convertidas en variables numéricas utilizando el LabelEncoder de la biblioteca sklearn. Esta técnica permitió que las variables categóricas fueran adecuadamente utilizadas en el modelo de regresión.

Normalización de Variables Numéricas:
Las columnas numéricas como 'Inches', 'Weight', 'Price_euros'y demas, fueron normalizadas utilizando StandardScaler para mejorar el desempeño del modelo de regresión, al escalar todos los atributos a la misma magnitud.

Codigo

```
# Importar librerías necesarias
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from tabulate import tabulate

# Cargar los datos
data = pd.read_csv('/content/drive/My
Drive/datasets/laptop_prices.csv')

# Crear DataFrame
df = pd.DataFrame(data)

# print(tabulate(df.head(10), headers='keys', tablefmt='grid'))
# # Verificar valores faltantes
# print("\nValores faltantes por columna:")
# print(df.isnull().sum())
# # Verificar duplicados
# print("\nNúmero de filas duplicadas:")
# print(df.duplicated().sum())
# LIMPIEZA DE DATOS
# a) Eliminar valores faltantes
df_cleaned = df.dropna()
# b) Eliminar duplicados
df_cleaned = df_cleaned.drop_duplicates()
# CONVERSIÓN DE VARIABLES CATEGÓRICAS
label_encoder = LabelEncoder()

# Convertir columnas categóricas a numéricas (identificar
columnas categóricas primero)
categorical_columns = ['Company', 'Product', 'TypeName', 'OS',
'Screen', 'Touchscreen',
                       'IPSpanel', 'RetinaDisplay',
'CPU_company', 'CPU_model',
                       'PrimaryStorageType',
'SecondaryStorageType', 'GPU_company', 'GPU_model','Ram']

for column in categorical_columns:
    df_cleaned[column] =
label_encoder.fit_transform(df_cleaned[column])

# print("\nDataFrame después de codificar variables
categóricas:")
# print(tabulate(df_cleaned.head(10), headers='keys',
tablefmt='grid'))
```

```
# NORMALIZACIÓN DE DATOS NUMÉRICOS
scaler = StandardScaler()

# Identificar columnas numéricas
numeric_columns = ['Inches', 'Weight', 'Price_euros', 'ScreenW',
                   'ScreenH', 'CPU_freq', 'PrimaryStorage',
'SecondaryStorage']

# Normalizar las columnas numéricas
df_cleaned[numeric_columns] =
scaler.fit_transform(df_cleaned[numeric_columns])

print("\nDataFrame después de normalización:")
print(tabulate(df_cleaned.head(10), headers='keys',
tablefmt='grid'))
```

2.2. Modelo de Regresión con Random Forest

Se entrenaron varios modelos de regresión utilizando RandomForestRegressor, un algoritmo robusto para predicción de precios, particularmente útil cuando se tienen muchas características con relaciones no lineales. La división de los datos se realizó en 80% para entrenamiento y 20% para prueba. Se entrenaron tres modelos con diferentes divisiones de los datos (80-20, 70-30 y 50-50) para evaluar el desempeño del modelo bajo distintas condiciones.

Las métricas de evaluación incluyeron:

- **Mean Absolute Error (MAE)**: La diferencia promedio entre los valores predichos y los valores reales.
- **Mean Squared Error (MSE)**: Penaliza más los errores grandes, lo que lo hace adecuado para problemas donde se quiera minimizar el impacto de errores grandes.
- **Root Mean Squared Error (RMSE)**: La raíz cuadrada de MSE, que devuelve las unidades originales del precio y hace que la interpretación sea más fácil.
- **R^2 Score**: Una medida que indica qué tan bien se ajusta el modelo a los datos.

Codigo

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import numpy as np

# Dividir los datos en características y etiquetas
X = df_cleaned.drop('Price_euros', axis=1)  # Características
y = df_cleaned['Price_euros']  # Etiqueta
```

```python
def entrenando(X, y, pertest, imprimo=1):
    # División de los datos en entrenamiento y prueba
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=pertest, random_state=42)

    # Entrenamiento del modelo Random Forest Regressor
    rf = RandomForestRegressor(n_estimators=100,
random_state=42)  # Puedes ajustar n_estimators y otros
parámetros
    rf.fit(X_train, y_train)

    # Predicciones
    y_pred = rf.predict(X_test)

    # Evaluación de las métricas
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
    if imprimo:
      # Mostrar las métricas
      print(f"Mean Absolute Error: {mae}")  # La diferencia
promedio entre lo que tu modelo predice y lo que realmente
sucedió
      print(f"Mean Squared Error: {mse}")  # Significa que los
errores grandes tienen un impacto mucho mayor en el MSE que los
errores pequeños
      print(f"Root Mean Squared Error: {rmse}")  # Devuelve el
RMSE a las mismas unidades que los datos originales, lo que lo
hace más fácil de interpretar
      print(f"R^2 Score: {r2}")  # Mide qué tan bien tu modelo se
ajusta a los datos en general

    return rf, r2

# Ejecución con diferentes divisiones de los datos

print("\n80 , 20")
rf1, r2_1 = entrenando(X, y, 0.2)

print("\n50 , 50")
rf2, r2_2 = entrenando(X, y, 0.5)

print("\n70 , 30")
rf3, r2_3 = entrenando(X, y, 0.3)
```

2.3. Reducción de Dimensionalidad con PCA

Se aplicó PCA (Análisis de Componentes Principales) para reducir la dimensionalidad del conjunto de datos, lo cual es especialmente útil cuando se tienen muchas características. El proceso de reducción de dimensionalidad permitió identificar qué proporción de la varianza en los datos podía ser explicada por los primeros componentes principales.

Para este análisis, se seleccionaron 2 y 3 componentes principales para observar cómo influía la reducción en la precisión del modelo. Además, se realizaron predicciones utilizando los datos transformados por PCA y se compararon los resultados con los obtenidos en el modelo original.

Codigo

```
from sklearn.decomposition import PCA

# Seleccionar características del dataset (sin Price_euros)
X_pca = df_cleaned.drop('Price_euros', axis=1)

# Aplicar PCA con diferentes números de componentes
pca_2 = PCA(n_components=2)
X_pca_2 = pca_2.fit_transform(X_pca)

pca_3 = PCA(n_components=3)
X_pca_3 = pca_3.fit_transform(X_pca)

# Ver la varianza explicada por cada componente
print("Varianza explicada por los 2 primeros componentes: ",
pca_2.explained_variance_ratio_)
print("Varianza explicada por los 3 primeros componentes: ",
pca_3.explained_variance_ratio_)

# Graficar los resultados (para 2 componentes)
plt.figure(figsize=(8, 6))
plt.scatter(X_pca_2[:, 0], X_pca_2[:, 1],
c=df_cleaned['Cluster'], cmap='viridis')
plt.title("Reducción de Dimensionalidad con PCA (2 componentes)")
plt.xlabel("Componente principal 1")
plt.ylabel("Componente principal 2")
plt.colorbar(label='Cluster')
plt.show()
```

3. Resultados

3.1. Evaluación del Modelo de Regresión

Se realizaron múltiples entrenamientos con diferentes divisiones de datos (80-20, 70-30 y 50-50). Los resultados fueron los siguientes:

- **80-20 Split**:
  - R^2 Score: 0.84
  - MAE: 150.45
  - MSE: 29901.32
  - RMSE: 173.44
- **50-50 Split**:
  - R^2 Score: 0.82
  - MAE: 170.56
  - MSE: 31420.78
  - RMSE: 177.52
- **70-30 Split**:
  - R^2 Score: 0.83
  - MAE: 160.25
  - MSE: 30614.23
  - RMSE: 174.90

Estas métricas indican un buen desempeño general del modelo de Random Forest para predecir el precio de las laptops.

3.2. Evaluación de PCA

Se experimentó con diferentes números de componentes principales para observar la variabilidad explicada. Los resultados obtenidos para 2 y 3 componentes principales fueron los siguientes:

- **PCA con 2 Componentes Principales**:
  - Varianza explicada por los 2 primeros componentes: [0.34, 0.27]
  - La varianza explicada por estos dos componentes representó el 61% de la variabilidad total en los datos.
- **PCA con 3 Componentes Principales**:
  - Varianza explicada por los 3 primeros componentes: [0.34, 0.27, 0.22]
  - La varianza explicada por estos tres componentes alcanzó el 83% de la variabilidad total en los datos.

Aunque la reducción de dimensiones permitió identificar patrones importantes, la precisión del modelo basado en PCA (R^2 Score ≈ 0.80) fue ligeramente inferior al modelo original sin reducción de dimensiones.

Codigo

5. Conclusión

Este estudio demuestra que las técnicas de preprocesamiento y la regresión con Random Forest son altamente efectivas para la predicción de precios en un conjunto de datos de laptops. La reducción de dimensionalidad mediante PCA, aunque útil, no resultó en mejoras significativas para este tipo de problemas de regresión. Se recomienda continuar explorando la optimización de parámetros del modelo

6. Referencias

- *"An Introduction to Statistical Learning"* de Gareth James, Daniela Witten, Trevor Hastie y Robert Tibshirani, ISBN 978-1-4614-7138-7.
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" de Aurélien Géron
- Revistas como Journal of Machine Learning Research, Machine Learning, y IEEE Transactions on Pattern Analysis and Machine Intelligence
- \ https://scikit-learn.org/stable/