Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Artificial Intelligence and Systems Engineering

Predicting Electricity Price Surges for Domestic Energy Trading

Student solution for Data Science Competition

Cuadros Rivas, Alejandra Paola

KK5459

Course: Machine Learning Use-case Laboratory

Field: Computer Engineering

Specialization: Data Science and Artificial Intelligence

2024/ Winter semester

Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics

Introduction

Electricity trading is a dynamic field that requires accurate predictions to maintain market efficiency and optimize trading strategies. This project addresses the challenge of predicting electricity price surges that significantly impact individual business consumers and producers. Specifically, the objective is to estimate, using historical data from the past four days, the hour on the fifth day when electricity prices will exceed the balancing energy price.

Data Exploration

The dataset, public_data.csv, contains a diverse range of features, including:

- Time-based Features: rowID, season, periodID, day_in_period, hour, minute, holyday, weekday
- Power Market Variables: ke, hupx, afrr_fel, afrr_le, mfrr_fel, mfrr_le, afrr
- Weather and Load Data: solar_becsult_dayahead, rendszerterheles_terv
- Target Variable: target_flag (binary indicator for price surge).

Data Preprocessing

The dataset was split into training and validation subsets using an 80/20 split ratio to ensure robust model evaluation:

```
X_train_split, X_val, y_train_split, y_val = train_test_split( X_train, y_train, test_size=0.2, random_state=42 )
```

Feature scaling was applied during Principal Component Analysis (PCA) to standardize the data, ensuring all features contributed equally to the dimensionality reduction:

```
scaler = StandardScaler()
X_train_pca = scaler.fit_transform(X_train)
```

PCA was used to reduce dimensionality, retaining key components for model training:

```
pca = PCA(n_components=10)
X_train_pca = pca.fit_transform(X_train)
```

Model Selection and Training

The Random Forest Classifier was selected as the model due to its robustness and capability to handle high-dimensional data effectively.

Initial parameters for the Random Forest Classifier included:

- n_estimators=100: Number of trees in the forest.
- max_depth=10: Maximum depth of each tree to control overfitting.
- min_samples_split=10: Minimum samples required to split a node.

- min_samples_leaf=5: Minimum samples required at a leaf node.

Model training was conducted on the training subset:

```
clf = RandomForestClassifier(
    random_state=42,
    n_estimators=100,
    max_depth=10,
    min_samples_split=10,
    min_samples_leaf=5
)
clf.fit(X_train_split, y_train_split)
```

Hyperparameter Tuning

The notebook demonstrates hyperparameter tuning by manually setting key parameters for a Random Forest Classifier, including:

- n_estimators=100: The number of trees in the forest. This value was fixed to ensure a balance between performance and computational cost.
- max_depth=10: Restricts the depth of each tree to prevent overfitting, particularly for small datasets.
- min_samples_split=10: Specifies the minimum number of samples required to split an internal node.
- min_samples_leaf=5: Ensures a minimum number of samples at each leaf node to prevent overfitting.

Model Evaluation

Two evaluation metrics were used to assess model performance:

1. ROC AUC (Area Under the Curve): Evaluates the model's ability to distinguish between positive and negative classes.
   validation_auc = roc_auc_score(y_val, y_val_pred_proba)
2. Log Loss: Measures the accuracy of predicted probabilities, with lower values indicating better calibration
   validation_log_loss = log_loss(y_val, y_val_pred_proba)

Validation metrics were calculated on the validation subset:

```
y_val_pred_proba = clf.predict_proba(X_val)[:, 1]
validation_auc = roc_auc_score(y_val, y_val_pred_proba)
validation_log_loss = log_loss(y_val, y_val_pred_proba)
print("Validation AUC:", validation_auc)
print("Validation Log Loss:", validation_log_loss)
```

Prediction and Submission

The trained model was used to predict probabilities for the test dataset:

```
y_test_pred_proba = clf.predict_proba(X_test)[:, 1]
```

Predictions were saved to a CSV file for submission, adhering to competition requirements:

```
submission = pd.DataFrame({'rowID': test_rowIDs, 'prediction': y_test_pred_proba})
submission.to_csv('Submission01.csv', index=False)
```

Multiple submissions were created, corresponding to different configurations, including the use of PCA-transformed data. The last values we obtained were Validation AUC: 0.9327358667791505 and Validation Log Loss: 0.3328956179231922.

Conclusion

This project effectively predicted electricity price surges using a well-optimized Random Forest Classifier, supported by thorough preprocessing, robust feature engineering, and advanced evaluation techniques, achieving high AUC scores. By addressing challenges such as overfitting and handling missing data, the model demonstrated strong performance and practical applicability in real-world electricity trading scenarios. Future improvements could include exploring alternative models, implementing advanced probability calibration, and further refining prediction accuracy and computational efficiency.