[År]

Case opgave i machine learning

H4 CASEOPGAVE

ALEXANDER JENSBY THOMSEN

Indhold

Introduktion

I denne case præsenteres en model, der skal forudsige fremtidens PPP (købekraftsparitet) for lande baseret på 31 forskellige indikatorer. Disse indikatorer er designet til at arbejde sammen med Èltetö-Köves-Szulc (EKS) metoden til beregning af PPP for produktgrupper (stadie 2) og aggregerede produkter (stadie 3). EKS-metoden er valgt, da datasættet stammer fra Eurostat, som selv anbefaler brug af denne metode til beregning af PPP.

Projektet anvender to modeller: Den første er en ensemble-læringsteknik, Random Forest Regression, som bruges til regression, og som er valgt på grund af dens nemme håndtering og skalerbarhed. Den anden model er LightGBM (Light Gradient Boosting Machine), en højtydende gradient boosting-algoritme, som er ideel til store datamængder.

Datasættet indeholder 61 finansielle indikatorer, som er opdelt i 4 kategorier af EKS. Tidsperioden strækker sig fra 1995 til 2023 (undtagen England), hvilket resulterer i et datasæt med 1.358.592 datafelter.

I projektet vil der være fokus på, hvordan modellerne teknisk fungerer, relevante kodespor samt en konklusion om modellernes præcision og anvendelighed.

Mål og problemformulering

Formålet med casen er at der skal udvikles en prædiktiv model, der kan forudsige fremtiden købekraft (PPP) for de forskellige lande i EU. Modellen Modellen vil blive bygget på baggrund af et datasæt bestående af 61 finansielle indikatorer, som er inddelt i 4 kategorier af Èltetö-Köves-Szulc (EKS) metoden.

Problemformulering

Hvordan kan maskinlæringsmodeller som Random Forest Regression og LightGBM anvendes til at forudsige købekraftsparitet (PPP) for forskellige lande i EU?

Metode og tilgang

Den første fase i processen var at finde et datasæt som kunne være tilstrækkeligt til formålet. Planen var hele tiden at Danmark skulle være med i dette Datasæt, så det var hurtigt at finde Eurostat, og deres meget store datasæt for landene i EU. I forhold til andre på holdet, forholdte jeg mig til at mit datasæt ikke var særligt stort, og derved var jeg klar til at finde en model som var god til et lille datasæt med en lineær regression. Dette skyldes at modellen skulle til at starte med kun forudsige et år af gangen, så der var ikke behov for mere komplekse modeller. Med andre ord skulle den kunne se hvordan det er gået fra start 2024 til slut 2024, altså en lineær linje fra en PPP værdi til en anden PPP værdi.

Til at starte med blev der valgt en model kaldet RandomForestRegressor. Grunden til dette skyldes at den er vel omtalt til relative små datasæt og dens nemme tilgang og skalering.

Efter at modellen var blevet valgt, er det blevet tid til at forberede data. Denne proces skulle vise sig at være meget vanskelig. Jeg startede med at tænke at den selv kunne udregne PPP værdierne ud fra indikatorerne, men forgæves, at dens endelige resultat ville være under 2.3% i præcision. Efter nermere undersøgelser, ændrede jeg taktik og derfor inkluderede jeg ikke kun indikatorerne men også de tidligere PPP værdier fra perioderne 1995 – 2023. Efter at sammensat datasættet korrekt sammen var jeg klar til at gå videre til træning af modellen.

```python
def improvement_data(indicator_data, ppp_data):
    print("Processing data...")
    country_mapping = {
        'AT': 'Austria', 'BE': 'Belgium', 'BG': 'Bulgaria', 'CH': 'Switzerland',
        'CY': 'Cyprus', 'CZ': 'Czechia', 'DE': 'Germany', 'DK': 'Denmark',
        'EE': 'Estonia', 'EL': 'Greece', 'ES': 'Spain', 'FI': 'Finland',
        'FR': 'France', 'HR': 'Croatia', 'HU': 'Hungary', 'IE': 'Ireland',
        'IS': 'Iceland', 'IT': 'Italy', 'LT': 'Lithuania', 'LU': 'Luxembourg',
        'LV': 'Latvia', 'MT': 'Malta', 'NL': 'Netherlands', 'NO': 'Norway',
        'PL': 'Poland', 'PT': 'Portugal', 'RO': 'Romania', 'SE': 'Sweden',
        'SI': 'Slovenia', 'SK': 'Slovakia', 'UK': 'United Kingdom'
    }

    indicator_data['country_name'] = indicator_data['geo'].map(country_mapping)
    indicator_data = indicator_data[indicator_data['country_name'] != 'United Kingdom']

    ppp_data_melted = ppp_data.melt(id_vars=['TIME'], var_name='TIME_PERIOD', value_name='PPP_VALUE')
    ppp_data_melted = ppp_data_melted[ppp_data_melted['TIME_PERIOD'] != 2021]

    indicator_data['TIME_PERIOD'] = indicator_data['TIME_PERIOD'].astype(int)
    ppp_data_melted['TIME_PERIOD'] = ppp_data_melted['TIME_PERIOD'].astype(int)

    print(f"Unique 'geo' in indicator_data: {indicator_data['geo'].unique()}")
    print(f"Unique 'TIME_PERIOD' in indicator_data: {indicator_data['TIME_PERIOD'].unique()}")
    print(f"Unique 'TIME' in ppp_data_melted: {ppp_data_melted['TIME'].unique()}")
    print(f"Unique 'TIME_PERIOD' in ppp_data_melted: {ppp_data_melted['TIME_PERIOD'].unique()}")

    combined_data = pd.merge(indicator_data, ppp_data_melted, left_on=['country_name', 'TIME_PERIOD'], right_on=['TIME', 'TIME_PERIOD'], how='inner')
    if combined_data.empty:
        print("Combined data is empty after merging.")
        return None, None

    train_data = combined_data[combined_data['TIME_PERIOD'] < 2019]
    test_data = combined_data[combined_data['TIME_PERIOD'] >= 2019]
```

Figur 1 metoden, der forbereder data til modellerne

Når data var mappet rigtigt, modulleret og forberedt, var det nu tid til at træne selve modellen. Her klassificere vi forskellige parameter som er med til at definere størrelsen på modellen. Dette ses i 'param_grid', hvor vi sætter bootstrap og max_depth.

```python
# Træn Random Forest-model
def train_RFG_model(train_data, test_data):
    # Forbered trænings- og testdata
    X_train = train_data.drop(columns=['PPP_VALUE', 'geo', 'STRUCTURE', 'STRUCTURE_ID', 'na_item', 'ppp_cat', 'OBS_FLAG', 'country_name'])
    y_train = train_data['PPP_VALUE']

    X_test = test_data.drop(columns=['PPP_VALUE', 'geo', 'STRUCTURE', 'STRUCTURE_ID', 'na_item', 'ppp_cat', 'OBS_FLAG', 'country_name'])
    y_test = test_data['PPP_VALUE']

    X_train = X_train.select_dtypes(include=['number'])  # Vælg kun numeriske kolonner
    X_test = X_test.select_dtypes(include=['number'])

    # Split træningsdata i træning og validation
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

    # Initialiser og tun model med hyperparameteroptimering
    model = RandomForestRegressor(random_state=42, n_jobs=-1)

    # Definer parametergrid for GridSearch
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [10, 20, 30, None],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'bootstrap': [True, False]
    }

    # GridSearch for at finde de bedste hyperparametre
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_  # Vælg bedste model

    print("Best parameters found: ", grid_search.best_params_)

    # Træn den bedste model
    best_model.fit(X_train, y_train)

    # Forudsig på testdata
    y_pred = best_model.predict(X_test)

    # Evaluer modellen
    return evaluate_model(y_test, y_pred)
```

Når modellen er trænet som ønsket, kan vi nu se på det resultat som vi ønsker. Resultatet er basseret på de evalueringsmetoder som man bruger til at se hvor godt en model klarer sig. Her vil det være de retunerede værdier som 'mse','mae',rmse','r2' og 'mape'. Alle er med til at evaluere modellen, og fortælle os omkring dens successrate og præcision. Her er det 'r2', som er den vigtigste. 'r2' er med til at fortælle omkring modellen samlede præcision, og er 0.32, eller 32% præcis.

```python
# Evaluer modellen og vis resultaterne
def evaluate_model(y_true, y_pred, country='Denmark'):
    mse = mean_squared_error(y_true, y_pred)  # Mean Squared Error
    mae = mean_absolute_error(y_true, y_pred)  # Mean Absolute Error
    rmse = np.sqrt(mse)  # Root Mean Squared Error
    r2 = r2_score(y_true, y_pred)  # R²-score
    mape = mean_absolute_percentage_error(y_true, y_pred)  # MAPE

    print(f"MSE: {mse}, MAE: {mae}, RMSE: {rmse}, R²: {r2}, MAPE: {mape}")

    # Hvis 'y_true' indeholder landene, filtrer kun for det ønskede land
    if isinstance(y_true, pd.Series) and 'country_name' in y_true.index.names:
        print(f"Filtering results for {country}")
        country_data = y_true.index.get_level_values('country_name') == country
        y_true = y_true[country_data]
        y_pred = y_pred[country_data]

    # Plot faktiske vs forudsigede værdier
    plt.figure(figsize=(8, 6))
    plt.plot(y_true.values, label='Actual', color='blue', alpha=0.7)
    plt.plot(y_pred, label='Predicted', color='orange', alpha=0.7)
    plt.xlabel("Index")
    plt.ylabel("PPP Value")
    title = f"Actual vs Predicted Values for {country}"
    plt.title(title)
    plt.legend()
    plt.show()

    # Plot residualer
    residuals = y_true - y_pred
    plt.figure(figsize=(8, 6))
    sns.histplot(residuals, kde=True, color='red', bins=30)
    plt.xlabel("Residuals")
    plt.title(f"Residuals Distribution for {country}")
    plt.show()

    return mse, mae, rmse, r2, mape  # Returnér evalueringsmålene
```

Figur 2 Evaluering af modellen

Konklusion

Efter at have evalueret modellen, er det tydeligt at se at modellens præcision, ikke lever op til den standard som kræves af modeller. Dette skyldes formodentligt valget af modellen. En anden model ville nok have været bedre egnet til jobbet. Dertil kunne parametrene også skaleres op, således at modllen havde mere tid til at regne, og måske havde en bedre forståelse af både datasættet og også resultatet af PPP værdierne.