# PRESENTATION DU PROJET 🚀

## Analyse et Prévision des Prix de l'Ethereum USD par la Modélisation ARIMA/SARIMA

Lath ESSOH et Laurence ADODO DAHOUE

# Brève description du projet!

## Etape 1: Récupération et visualisation des données

# code!

```
import yfinance as yf

# Définition des dates de début et de fin pour la récupération des données
    today = date.today()
    end_date = today.strftime("%Y-%m-%d")
    start_date = (date.today() - timedelta(days=120)).strftime("%Y-%m-%d")
```

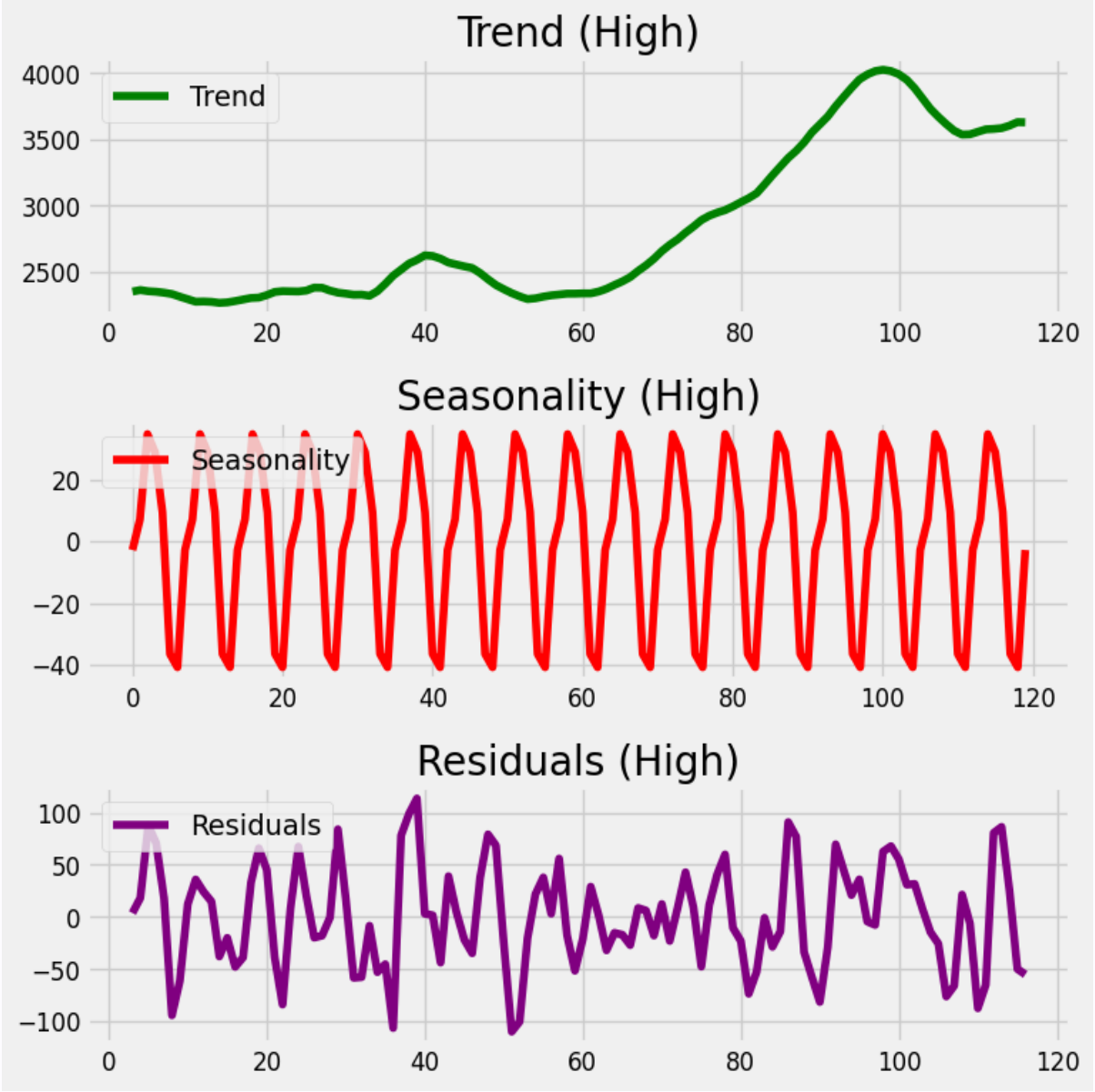
```
data = yf.download('ETH-USD', start=start_date, end=end_date, progress=False)
```

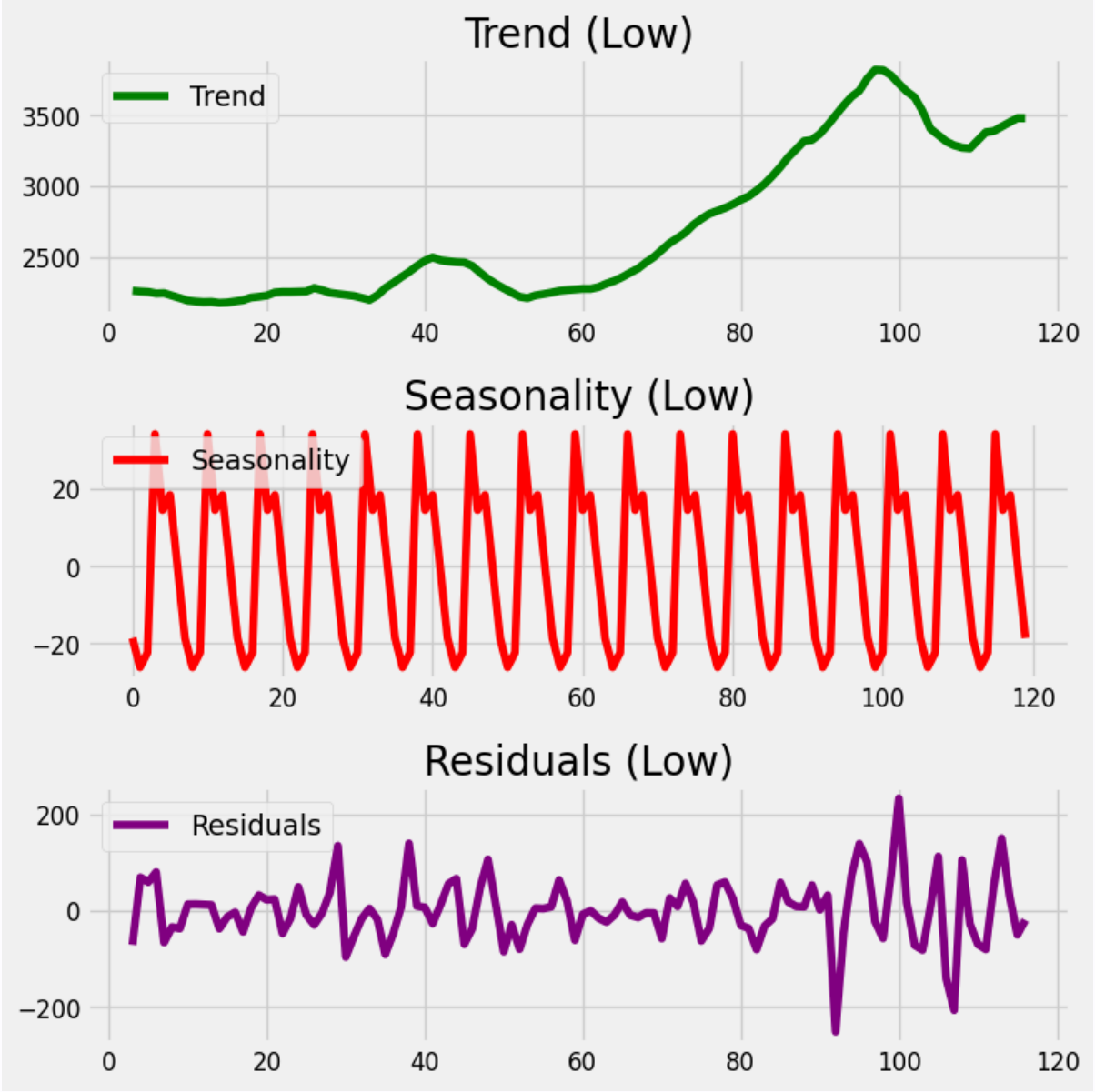
## Etape 2: Extraction des prix low et high

# code!

```
import yfinance as yf

crypto_prices_high = data.set_index('Date')['High']
crypto_prices_low = data.set_index('Date')['Low']
print(crypto_prices_high.head())
print(crypto_prices_low.head())
```


Trend (High)
Trend
4000
3500
3000
2500
0
20
40
60
80
100
120
Seasonality (High)
Seasonality
20
0
−20
−40
0
20
40
60
80
100
120
Residuals (High)
Residuals
100
50
0
−50
−100
0
20
40
60
80
100
120


Trend (Low)
Trend
3500
3000
2500
0
20
40
60
80
100
120
Seasonality (Low)
Seasonality
20
0
−20
0
20
40
60
80
100
120
Residuals (Low)
Residuals
200
0
−200
0
20
40
60
80
100
120

## Etape 3: Création du modèle automatique

# code!

```
from pmdarima import auto_arimaend=end_date, progress=False)
# Créer le modèle SARIMA automatique pour les prix hauts (High)
model_auto_high = auto_arima(crypto_prices_high, seasonal=True, m=12, trace=True)

# Créer le modèle SARIMA automatique pour les prix bas (Low)
model_auto_low = auto_arima(crypto_prices_low, seasonal=True, m=12, trace=True)
```

## Etape 4: Prédictions pour les dix prochains jours

# code!

```
predictions_sarima_high, conf_int_high = model_auto_high.predict_in_sample(return_conf_int=True)
predictions_sarima_low, conf_int_low = model_auto_low.predict_in_sample(return_conf_int=True)

all_predictions_sarima_high = pd.concat([pd.Series(predictions_sarima_high, index=crypto_prices_high.index)], axis=0)
all_predictions_sarima_low = pd.concat([pd.Series(predictions_sarima_low, index=crypto_prices_low.index)], axis=0)

forecast_sarima_high = model_auto_high.predict(start=len(crypto_prices_high), end=len(crypto_prices_high) + 10)
forecast_sarima_low = model_auto_low.predict(start=len(crypto_prices_low), end=len(crypto_prices_low) + 10)

index_future_dates = pd.date_range(start=crypto_prices_high.index[-1], periods=11, freq='D')[1:]
forecast_sarima_high_df = pd.DataFrame(forecast_sarima_high, index=index_future_dates, columns=['Predictions (High)'])

forecast_sarima_low_df = pd.DataFrame(forecast_sarima_low, index=index_future_dates, columns=['Predictions (Low)'])
```

# Etape 5: Création du PDF

# code!

```
from fpdf import FPDF
import matplotlib.pyplot as plt

# Ajout des images au PDF
    pdf.image('predictions_high.png', x=10, y=pdf.get_y(), w=180)
    pdf.ln(100)
    pdf.image('predictions_low.png', x=10, y=pdf.get_y(), w=180)
    pdf.ln(100)

    # Ajout des valeurs numériques au PDF
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="Valeurs numériques des prédictions pour les prix [Low ; High] :", ln=True, align='L')
    pdf.ln(5)

    for date, high_price, low_price in zip(forecast_sarima_high_df.index, forecast_sarima_high_df['Predictions (High)'], forecast_sarima_low_df['Predictions (Low)']):
        pdf.cell(200, 10, txt=f"{date.strftime('%Y-%m-%d')} : [{low_price}; {high_price}]", ln=True)

    pdf.output("predictions.pdf")
```

## Etape 6: Envoi du mail

- Création de l'objet MIMEMultipart: Cela crée un objet pour représenter un message MIME multipart, qui est utilisé pour les emails avec pièces jointes.
- Définition des en-têtes du message: Les champs comme "From", "To" et "Subject" sont définis.
- Corps du message: Le corps de l'email est attaché au message.

- Pièce jointe : fichier PDF : Le fichier PDF contenant les prévisions est ouvert en mode lecture binaire et attaché au message.
- Encodage de la pièce jointe : La pièce jointe est encodée en base64 pour être ajoutée au message.
- Ajout de l'en-tête de la pièce jointe : L'en-tête Content-Disposition est ajouté pour spécifier le nom de la pièce jointe.
- Envoi du courriel : Une connexion SSL est établie avec le serveur SMTP de Gmail, puis le message est envoyé avec les informations d'identification du compte expéditeur.

## Etape 7: Boucle pour l'envoi chaque 24h

```
import time

# Définir la fréquence de mise à jour en secondes
update_frequency = 86400  # Mettre à jour toutes les 24 heures

while True:
    try:

    time.sleep(update_frequency)
```

## Etape 8: Utilisation et exécution de l’outil

# VOIR README!