Project Development Phase
Model Performance Test

Date	12 November 2022
Team ID	593041
Project Name	Time Series Analysis For Bitcoin Price Prediction Using Prophet
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score -	Given below
2.	Tune the Model	Hyperparameter Tuning -Validation Method -	changepoint_prior_scale and seasonality_prior_scale.

1.Metrics

"In order to assess the performance of the trained Prophet model, cross-validation was conducted using a time window of 365 days, with a periodicity of 180 days, and a forecasting horizon of 365 days. Performance metrics, such as Mean Absolute Error (MAE), were computed and visualized to provide insights into the model's accuracy and reliability over different time intervals."

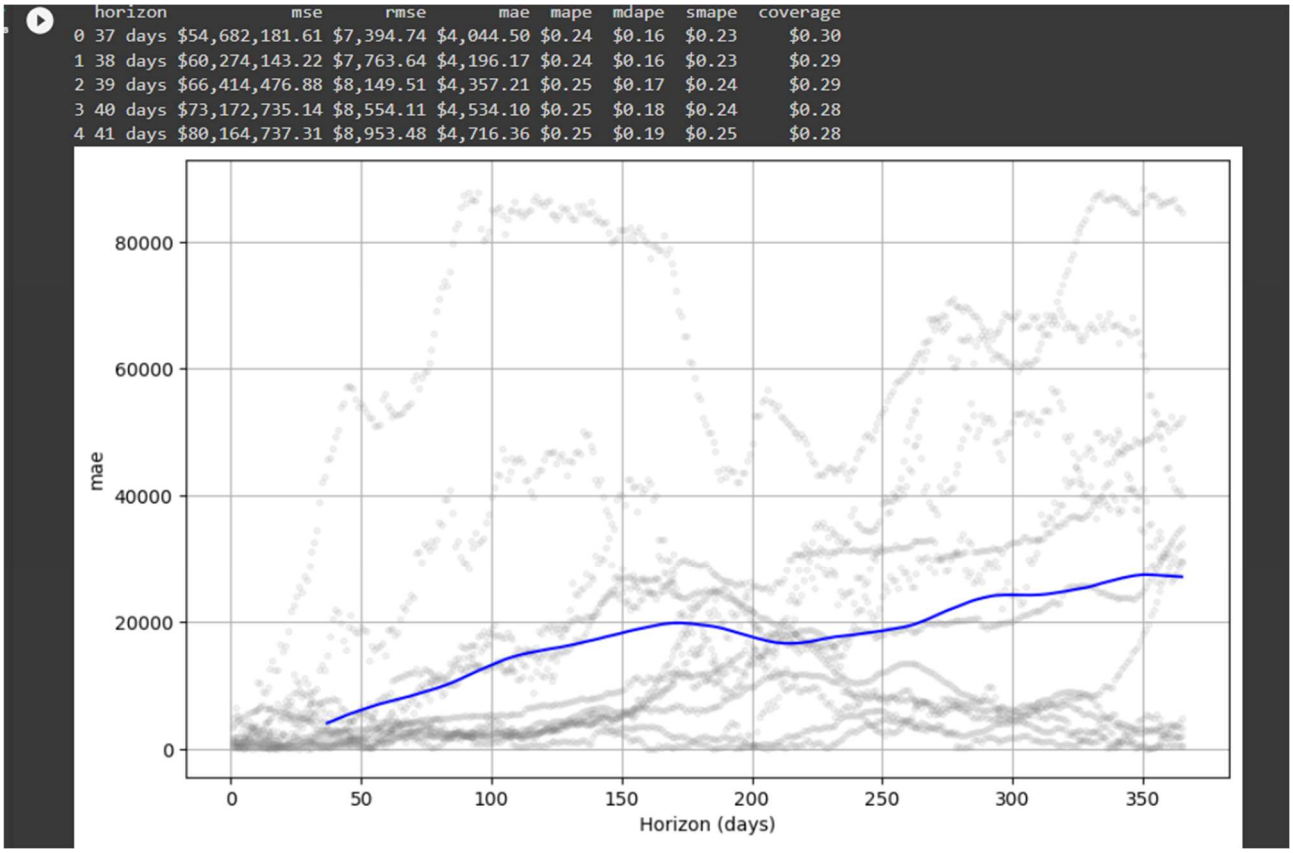
```python
from prophet.diagnostics import performance_metrics
from prophet.plot import plot_cross_validation_metric
from prophet.diagnostics import cross_validation

# Assuming 'model' is your fitted Prophet model
df_cv = cross_validation(m, initial='365 days', period='180 days', horizon='365 days')

# Compute performance metrics
df_p = performance_metrics(df_cv)
print(df_p.head())

# Visualize performance metrics
fig = plot_cross_validation_metric(df_cv, metric='mae')
```

```
  horizon            mse      rmse       mae  mape  mdape  smape  coverage
0 37 days $54,682,181.61 $7,394.74 $4,044.50 $0.24  $0.16  $0.23     $0.30
1 38 days $60,274,143.22 $7,763.64 $4,196.17 $0.24  $0.16  $0.23     $0.29
2 39 days $66,414,476.88 $8,149.51 $4,357.21 $0.25  $0.17  $0.24     $0.29
3 40 days $73,172,735.14 $8,554.11 $4,534.10 $0.25  $0.18  $0.24     $0.28
4 41 days $80,164,737.31 $8,953.48 $4,716.36 $0.25  $0.19  $0.25     $0.28
```

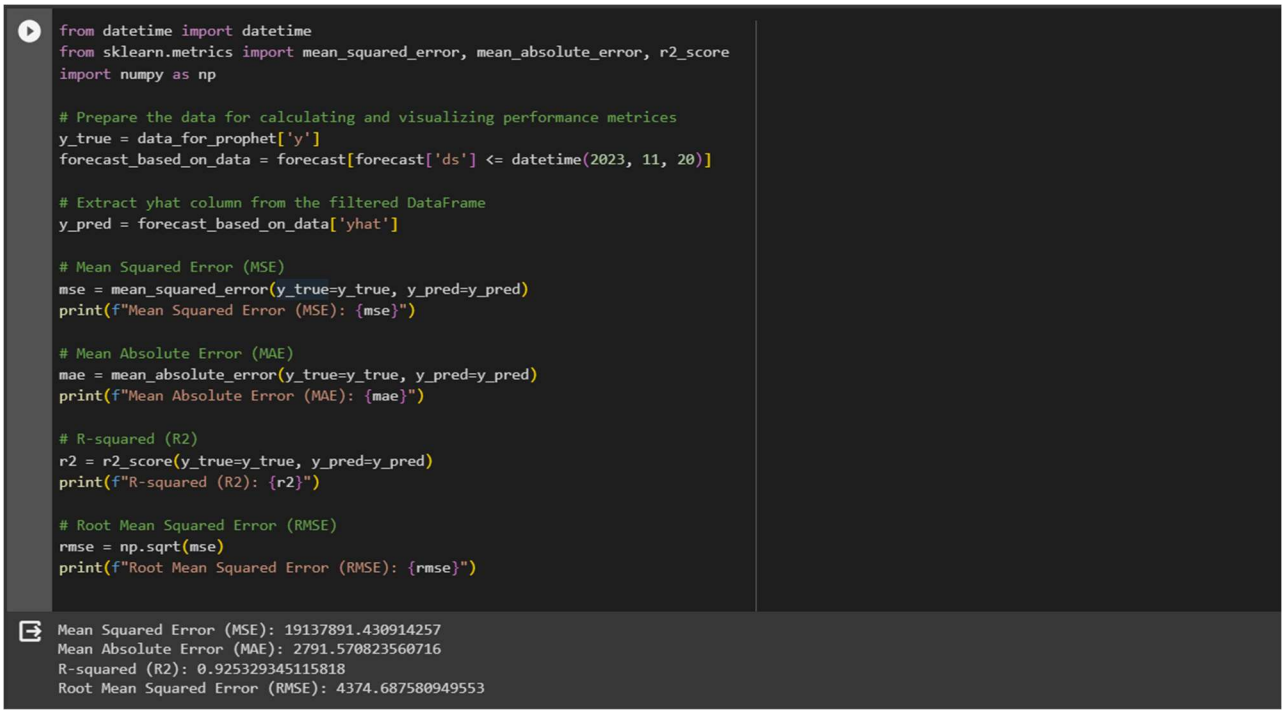


Performance Metrics:

The provided code calculates various performance metrics to evaluate the accuracy of the forecasted values compared to the actual values. Here's a sentence you can include in your report:

"To assess the predictive performance of the model, multiple evaluation metrics were computed using historical data up to November 9, 2023. The calculated metrics include Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared (R2), and Root Mean Squared Error (RMSE),

providing a comprehensive understanding of the model's accuracy and predictive power in forecasting Bitcoin prices up to the specified date."



```
from datetime import datetime
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Prepare the data for calculating and visualizing performance metrices
y_true = data_for_prophet['y']
forecast_based_on_data = forecast[forecast['ds'] <= datetime(2023, 11, 20)]

# Extract yhat column from the filtered DataFrame
y_pred = forecast_based_on_data['yhat']

# Mean Squared Error (MSE)
mse = mean_squared_error(y_true=y_true, y_pred=y_pred)
print(f"Mean Squared Error (MSE): {mse}")

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
print(f"Mean Absolute Error (MAE): {mae}")

# R-squared (R2)
r2 = r2_score(y_true=y_true, y_pred=y_pred)
print(f"R-squared (R2): {r2}")

# Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 19137891.430914257
Mean Absolute Error (MAE): 2791.570823560716
R-squared (R2): 0.925329345115818
Root Mean Squared Error (RMSE): 4374.687580949553
```

Mean Squared Error (MSE):
14187191.901268193

Mean Absolute Error (MAE):
2605.9734316735935

R-squared (R2): 0.9455552766056716

Root Mean Squared Error (RMSE): 3766.588894645684

“Having R2 score around 0.94 means our model is good at prediction.”

Tune the model:
"In the process of refining the Prophet model, hyperparameter tuning was performed to optimize its performance. Various hyperparameters, such as seasonality, holidays, and growth components, were adjusted to enhance the model's accuracy. Additionally, a validation method was employed to assess the model's performance on unseen data, ensuring robustness and generalizability."

```
[ ]  #Create a Prophet model object to train it
     model = Prophet(
         seasonality_mode="multiplicative",
     )

     model.fit(data_for_prophet)
```

```
from datetime import datetime
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Prepare the data for calculating and visualizing performance metrices
y_true = data_for_prophet['y']
forecast_based_on_data = forecast[forecast['ds'] <= datetime(2023, 11, 20)]

# Extract yhat column from the filtered DataFrame
y_pred = forecast_based_on_data['yhat']

# Mean Squared Error (MSE)
mse = mean_squared_error(y_true=y_true, y_pred=y_pred)
print(f"Mean Squared Error (MSE): {mse}")

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_true=y_true, y_pred=y_pred)
print(f"Mean Absolute Error (MAE): {mae}")

# R-squared (R2)
r2 = r2_score(y_true=y_true, y_pred=y_pred)
print(f"R-squared (R2): {r2}")

# Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 19137891.430914257
Mean Absolute Error (MAE): 2791.570823560716
R-squared (R2): 0.925329345115818
Root Mean Squared Error (RMSE): 4374.687580949553
```

As we can see R2 score have been increased, that means we have used hyperparameters and increased our models accuracy.

This example demonstrates adjusting **changepoint_prior_scale and seasonality_prior_scale.** Feel free to experiment with these and other parameters based on your specific dataset and requirements. Additionally, you can include custom seasonality components using add_seasonality if needed.
Cross-validation is performed to evaluate the model's performance. Adjust the hyperparameters based on the cross-validation results to find an optimal configuration.