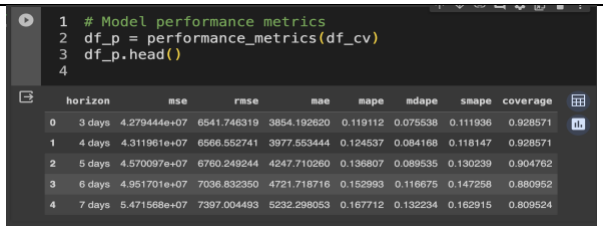
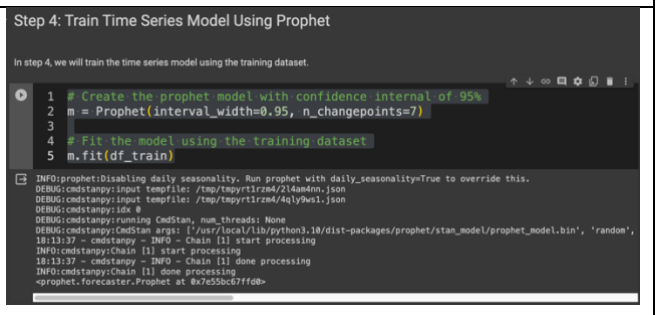
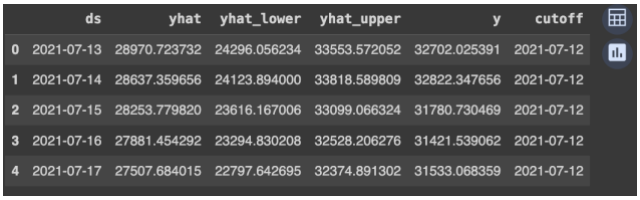
Project Development Phase
Model Performance Test

Date	9 November 2022
Team ID	Team-592401
Project Name	Time Series Analysis For Bitcoin Price Prediction Using Prophet
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - 4247.710260(5 days) , MSE - 4.570097e+07(5 days) , RMSE – 6760.249244 (5 days) , R2 score - 0.06465503430782771	 1 # Model performance metrics 2 df_p = performance_metrics(df_cv) 3 df_p.head() 4 horizon mse rmse mae mape mdape smape coverage 0 3 days 4.279444e+07 6541.746319 3854.192620 0.119112 0.075538 0.111936 0.928571 1 4 days 4.311961e+07 6566.552741 3977.553444 0.124537 0.084168 0.118147 0.928571 2 5 days 4.570097e+07 6760.249244 4247.710260 0.136807 0.089535 0.130239 0.904762 3 6 days 4.951701e+07 7036.832350 4721.718716 0.152993 0.116675 0.147258 0.880952 4 7 days 5.471568e+07 7397.004493 5232.298053 0.167712 0.132234 0.162915 0.809524  [27] 1 r2_score(df_cv.y, df_cv.yhat) 0.06465503430782771

2.	Tune the Model	Hyperparameter Tuning –
		# Create the prophet model with confidence internal of 95%
		m = Prophet(interval_width=0.95, n_changepoints=7)
		
		# Fit the model using the training dataset
		m.fit(df_train)
		
		Validation Method –
		In step 8, we will do cross-validation for the time series model. Prophet has a cross_validation function to automate the comparison between the actual and the predicted values.
			
			Step 4: Train Time Series Model Using Prophet
			In step 4, we will train the time series model using the training dataset.
			1 # Create the prophet model with confidence internal of 95%
			2 m = Prophet(interval_width=0.95, n_changepoints=7)
			3
			4 # Fit the model using the training dataset
			5 m.fit(df_train)
			INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
			DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/2l4am4nn.json
			DEBUG:cmdstanpy:input tempfile: /tmp/tmpyrt1rzm4/4qly9ws1.json
			DEBUG:cmdstanpy:idx 0
			DEBUG:cmdstanpy:running CmdStan, num_threads: None
			DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_model/prophet_model.bin', 'random',
			18:13:37 - cmdstanpy - INFO - Chain [1] start processing
			INFO:cmdstanpy:Chain [1] start processing
			18:13:37 - cmdstanpy - INFO - Chain [1] done processing
			INFO:cmdstanpy:Chain [1] done processing
			<prophet.forecaster.Prophet at 0x7e55bc67ffd0>
			
			 
			Step 8: Cross Validation
			In step 8, we will do cross-validation for the time series model. Prophet has a cross_validation function to automate the comparison between the actual and the predicted values.
			• 'm' is the trained model.
			• 'initial='500 days'' means the initial model will be trained on the first 500 days of data.
			• 'period='60 days'' means 60 days will be added to the training dataset for each additional model.
			• horizon = '30 days' means that the model forecasts the next 30 days. When only horizon is given, Prophet defaults initial to be triple the horizon, and period to be half of the horizon.
			• parallel="processes" enables parallel processing for cross-validation. When the parallel cross-validation can be done on a single machine, "processes" provide the highest performance. For larger problems, "dask" can be used to do cross-validation on multiple machines.
			1 # Cross validation
			2 df_cv = cross_validation(m, initial='500 days', period='60 days', horizon = '30 days', parallel="processes")
			3 df_cv.head()

		• `m' is the trained model. • `initial='500 days" means the initial model will be trained on the first 500 days of data. • `period='60 days" means 60 days will be added to the training dataset for each additional model. • horizon = '30 days' means that the model forecasts the next 30 days. When only horizon is given, Prophet defaults initial to be triple the horizon, and period to be half of the horizon. • parallel="processes" en ables parallel processing for cross-validation. When the parallel cross-validation can be done on a single machine, "processes" provide the highest performance. For larger problems, "dask" can be used to do cross-validation on multiple machines.	

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2021-07-13	28970.723732	24296.056234	33553.572052	32702.025391	2021-07-12
1	2021-07-14	28637.359656	24123.894000	33818.589809	32822.347656	2021-07-12
2	2021-07-15	28253.779820	23616.167006	33099.066324	31780.730469	2021-07-12
3	2021-07-16	27881.454292	23294.830208	32528.206276	31421.539062	2021-07-12
4	2021-07-17	27507.684015	22797.642695	32374.891302	31533.068359	2021-07-12