The Derek Zoolander Center for Machines Who Can’t Read Good and Wanna Learn to Do Other Stuff Good Too

Norman Adkins
Jae Park
David Dam
Louis Cheng

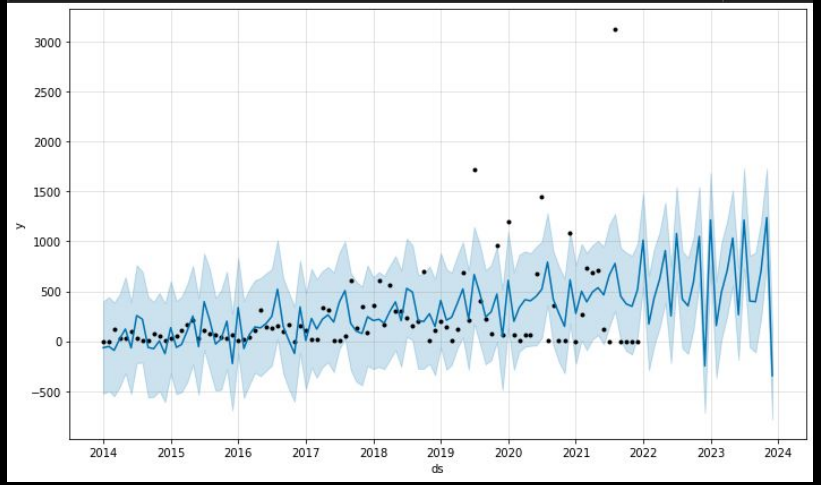
Data and Code

David Dam

	SOrg.	DChl	Distribution Channel	Sold-to pt	Ship-to	Plnt	Matl Group	Material Group	Material	Month	Invoiced Quantity
0	10	11	E-Store	1025073	1025073	10	300	NaN	TB74255100	01/2014	2
1	10	11	E-Store	1025073	1025073	10	300	NaN	TB72683000	01/2014	2
2	10	11	E-Store	1025073	1025073	10	300	NaN	TB72683100	01/2014	2
3	10	11	E-Store	1025073	1025073	10	300	NaN	TB35858000	01/2014	2
4	10	11	E-Store	1025073	1025073	10	300	NaN	IB25400000	01/2014	2
...	...	...	...	...	...	...	...	...	...	...	...
87018	10	20	Bicycle	1035929	1035929	30	300	NaN	SIB6988610	12/2021	1
87019	10	20	Bicycle	1037697	1037697	30	300	NaN	TB74180100	12/2021	50
87020	10	20	Bicycle	2000030	2000030	30	300	NaN	TB85975400	12/2021	5
87021	10	20	Bicycle	2000030	2000030	30	300	NaN	TB00184900	12/2021	3
87022	10	20	Bicycle	2000030	2000030	30	300	NaN	TB00096900	12/2021	5

87023 rows × 12 columns

```
# model with best hyperameters
m = Prophet(seasonality_mode=sm, changepoint_prior_scale=cps, seasonality_prior_scale=sps)

m.fit(df)
future = m.make_future_dataframe(periods=24, freq='M')

fcst_tuned = m.predict(future)
```



Summary of the Data

Actual sales data of my company's bicycle products from 2014 through 2021 extracted as a CSV from our ERP software.

Columns were significantly reduced to remove sensitive information that was outside of the scope of this project to maintain confidentiality.

The relevant data for this project is primarily the material, month, and invoiced quantity to perform a time series analysis.

	SOrg.	DChl	Distribution Channel	Sold-to pt	Ship-to	Plnt	Matl Group	Material Group	Material	Month	Invoiced Quantity
0	10	11	E-Store	1025073	1025073	10	300	NaN	TB74255100	01/2014	2
1	10	11	E-Store	1025073	1025073	10	300	NaN	TB72683000	01/2014	2
2	10	11	E-Store	1025073	1025073	10	300	NaN	TB72683100	01/2014	2
3	10	11	E-Store	1025073	1025073	10	300	NaN	TB35858000	01/2014	2
4	10	11	E-Store	1025073	1025073	10	300	NaN	IB25400000	01/2014	2
...	...	...	...	...	...	...	...	...	...	...	...
87018	10	20	Bicycle	1035929	1035929	30	300	NaN	SIB6988610	12/2021	1
87019	10	20	Bicycle	1037697	1037697	30	300	NaN	TB74180100	12/2021	50
87020	10	20	Bicycle	2000030	2000030	30	300	NaN	TB85975400	12/2021	5
87021	10	20	Bicycle	2000030	2000030	30	300	NaN	TB00184900	12/2021	3
87022	10	20	Bicycle	2000030	2000030	30	300	NaN	TB00096900	12/2021	5

87023 rows × 12 columns

The Code

Pandas was utilized to read the CSV and transform the data by grouping by month and material and aggregating the invoiced quantity, then to match Prophet's required input format.

```
[ ]  # pivot sold to and material
     sales_pivot_df = pd.pivot_table(sales_gb_df,
                                     values = 'Invoiced Quantity',
                                     index = 'Month',
                                     columns = 'Material',
                                     aggfunc=np.sum,
                                     fill_value = 0
                                          )

     sales_pivot_df = sales_pivot_df.reset_index()
     sales_pivot_df['Month'] = to_datetime(sales_pivot_df['Month'])
     sales_pivot_df = sales_pivot_df.set_index(['Month'])

     print(sales_pivot_df)
```

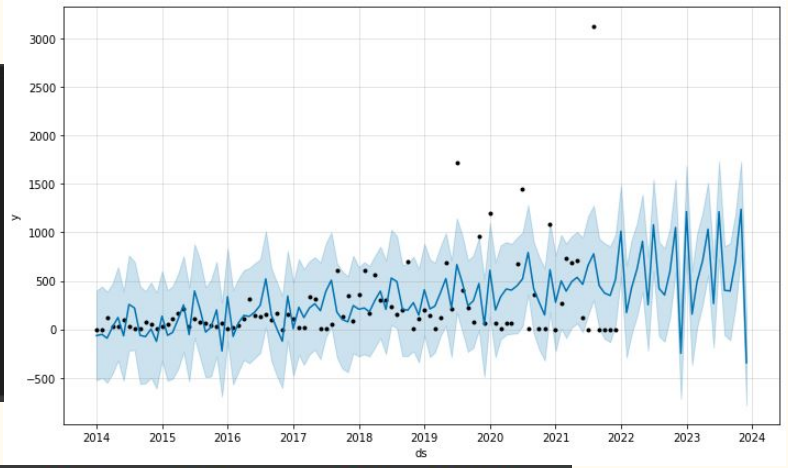
Material	TB96910100	TB96962000	TB96962100
Month			
2014-01-01	0	0	0
2014-02-01	0	0	0
2014-03-01	0	0	0
2014-04-01	0	0	0
2014-05-01	0	0	0
...	...	...	...
2021-08-01	0	42	0
2021-09-01	0	0	0
2021-10-01	0	0	15
2021-11-01	0	73	13
2021-12-01	45	0	0

[96 rows x 939 columns]

The Code

Prophet is able to fit non-linear trends with seasonal effects and shifts in trends and was used to model the sales data and forecast demand over the next 24 months to match lead times for the purpose of inventory/production planning.

```
# fit prophet model and predict
m = Prophet()

m.fit(df)
future = m.make_future_dataframe(periods=24, freq='M')

fcst = m.predict(future)
```



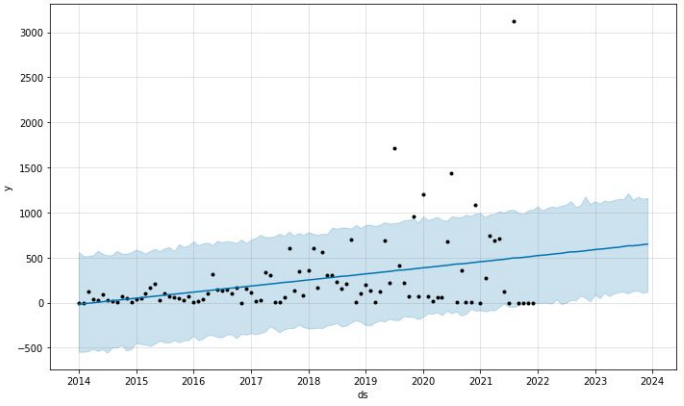
	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	730 days	149508.054335	386.662714	236.930151	3.141955	0.540087	0.714955	0.416667

The Code

Prophet is able to fit non-linear trends with seasonal effects and shifts in trends and was used to model the sales data and forecast demand over the next 24 months to match lead times for the purpose of inventory/production planning. Cross-validation was performed to measure performance and tune hyperparameters.

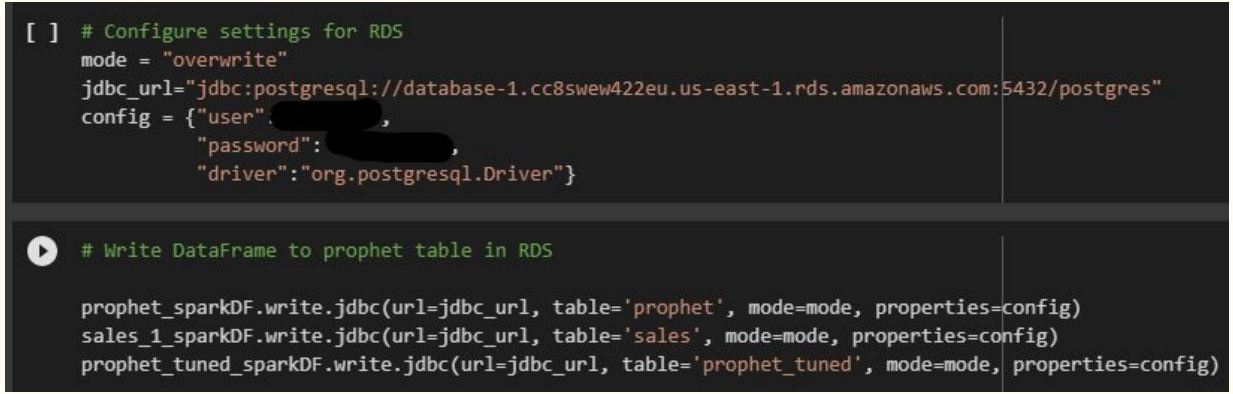
```
changepoint_prior_scale        0.005
seasonality_prior_scale         10.0
seasonality_mode             additive
growth                       logistic
rmse                        367.05949
Name: 14, dtype: object
```

```
[82] # model with best hyperameters
     m = Prophet(seasonality_mode=sm, changepoint_prior_scale=cps, seasonality_prior_scale=sps)

     m.fit(df)
     future = m.make_future_dataframe(periods=24, freq='M')

     fcst_tuned = m.predict(future)
```



The Code

Spark was then utilized to load the results to RDS.



```
[ ]  # Configure settings for RDS
     mode = "overwrite"
     jdbc_url="jdbc:postgresql://database-1.cc8swew422eu.us-east-1.rds.amazonaws.com:5432/postgres"
     config = {"user":[redacted],
               "password":[redacted],
               "driver":"org.postgresql.Driver"}
```

```
     # Write DataFrame to prophet table in RDS

     prophet_sparkDF.write.jdbc(url=jdbc_url, table='prophet', mode=mode, properties=config)
     sales_1_sparkDF.write.jdbc(url=jdbc_url, table='sales', mode=mode, properties=config)
     prophet_tuned_sparkDF.write.jdbc(url=jdbc_url, table='prophet_tuned', mode=mode, properties=config)
```

Architecture

Norman Adkins

Backend

Leveraged Amazon Web Services (AWS) RDS using PostgreSQL:

- Given the clear need of performing joins, a relational database was necessary.
- PostgreSQL was used because of familiarity and all features are open-source
- We opted to use AWS' Relational Database Service (RDS) because:
  - We wanted the data to be accessible to users without the constraints of an UI
  - We had an open instance spun up and available for this data
  - There were no costs for using this service due to the limited service requirements

Front-end

Ultimately, we decided to use Tableau as our front-end technology:

- There was a need to manipulate visuals as the model was being tuned
- Additional tables were added as the model was being tuned.
  - Tableau instantly picks those changes
- We needed to replicate visuals across both the test, training, and tuning data
  - Tableau offers the ability to quickly replicate this through features.
- The team had Tableau licenses available for use
- Tableau public offers the ability to publish the data.

Constraint: Real-time collaboration on the visuals.

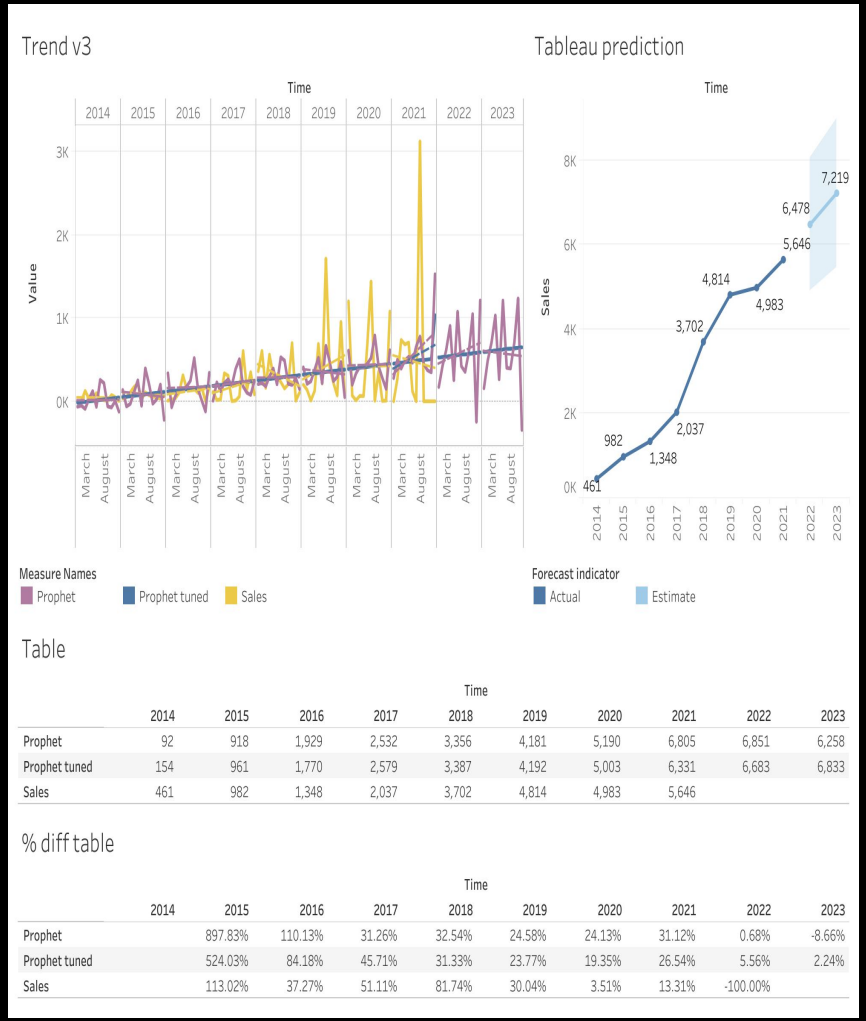
From Source-to-User

Raw Data Format	Data Cleansing and Preprocessing	Machine Learning and Analysis	Extract, Transform, and Load (ETL)	Database (Live Connection to UI)	Database (Live Connection to DB)
CSV	NumPy pandas	PROPHET matplotlib	pandas PySpark	PostgreSQL Amazon RDS	tableau A Salesforce Company

Import Raw Data, Clean, and Preprocess → Machine Learning and Analysis → Extract, Transform, and Load → Database Repository (AWS RDS) → Data Visualization / User Interface

Visualizations

Jae Park & Louis Cheng



Table

	Time									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Prophet	92	918	1,929	2,532	3,356	4,181	5,190	6,805	6,851	6,258
Prophet tuned	154	961	1,770	2,579	3,387	4,192	5,003	6,331	6,683	6,833
Sales	461	982	1,348	2,037	3,702	4,814	4,983	5,646		

% diff table

	Time									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Prophet		897.83%	110.13%	31.26%	32.54%	24.58%	24.13%	31.12%	0.68%	-8.66%
Prophet tuned		524.03%	84.18%	45.71%	31.33%	23.77%	19.35%	26.54%	5.56%	2.24%
Sales		113.02%	37.27%	51.11%	81.74%	30.04%	3.51%	13.31%	-100.00%	

Forecasting with Facebook Prophet

- Normal years
- Covid years
- Future

Table

	Time									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Prophet	92	918	1,929	2,532	3,356	4,181	5,190	6,805	6,851	6,258
Prophet tuned	154	961	1,770	2,579	3,387	4,192	5,003	6,331	6,683	6,833
Sales	461	982	1,348	2,037	3,702	4,814	4,983	5,646		

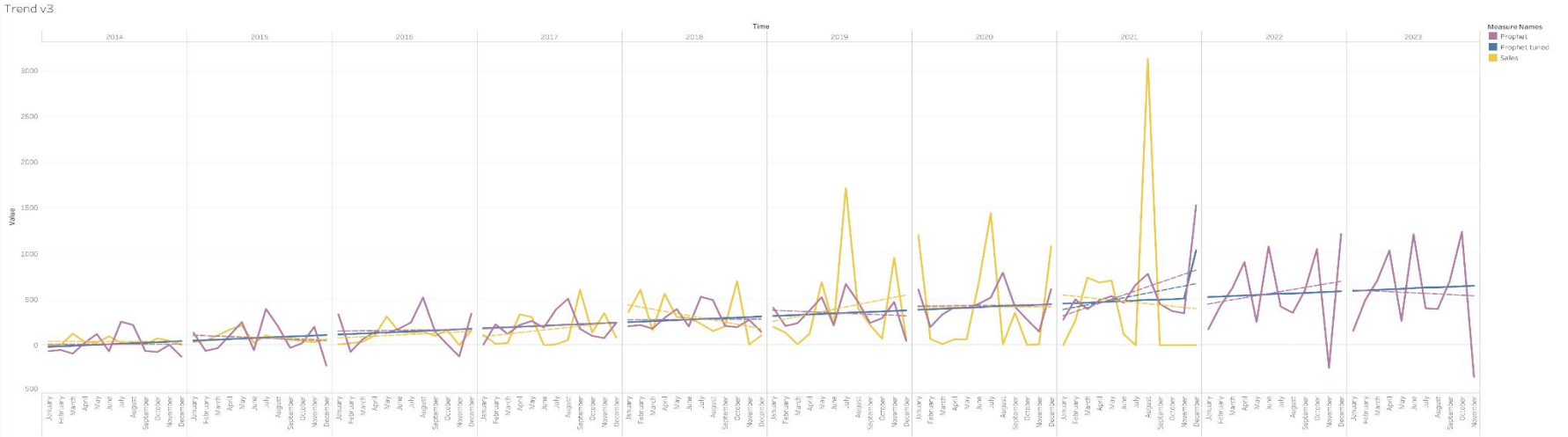
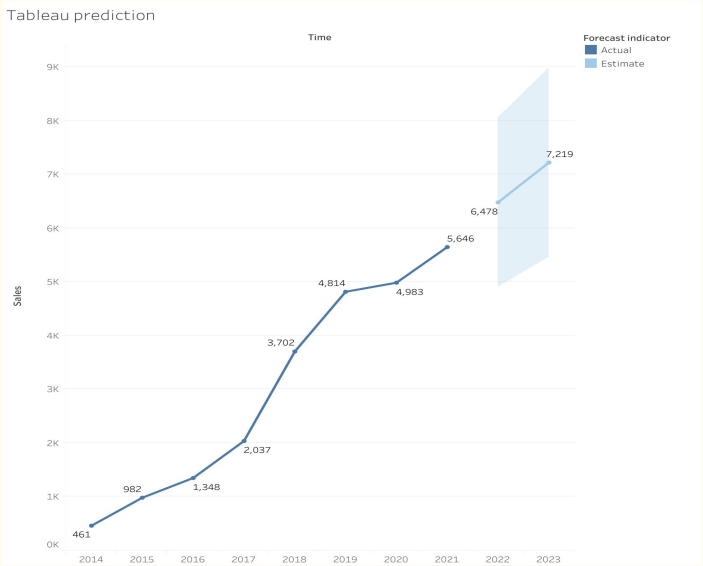


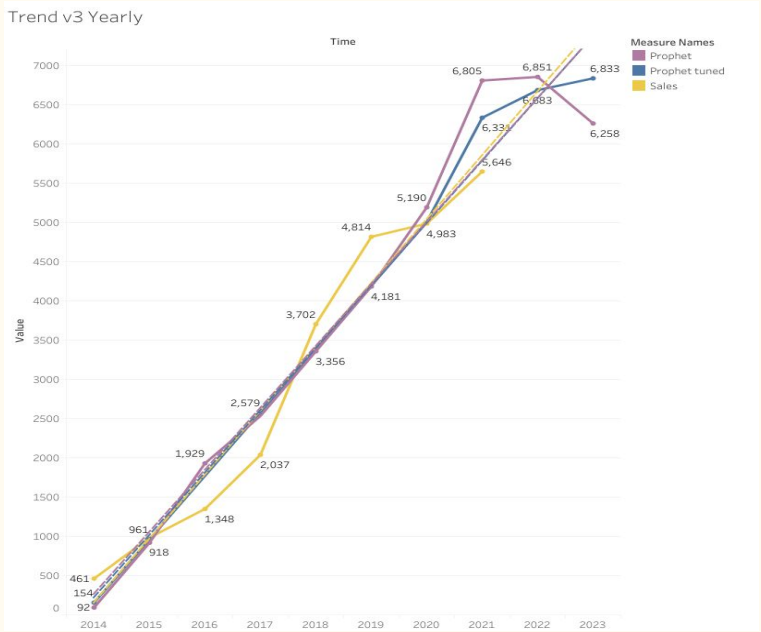
Tableau built in forecasting

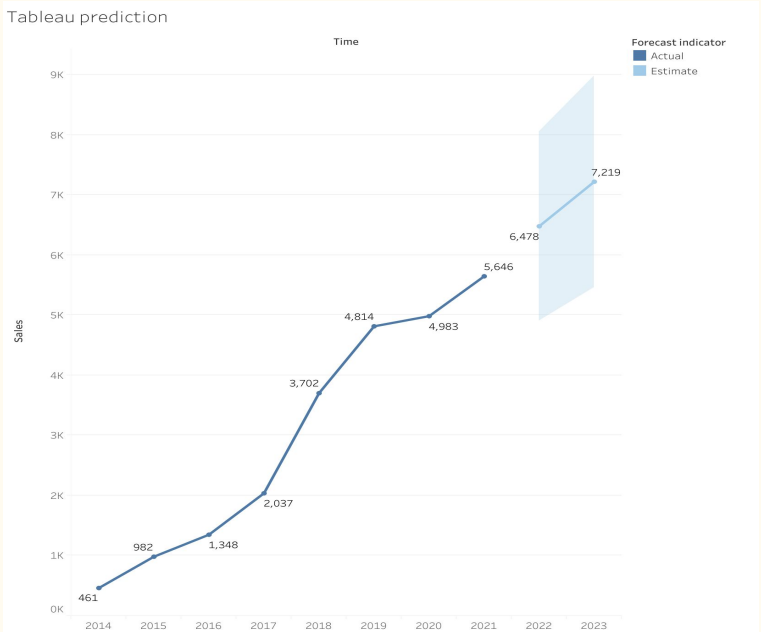
Uses Exponential Triple Smoothing (aka Holt-Winters) forecasting model



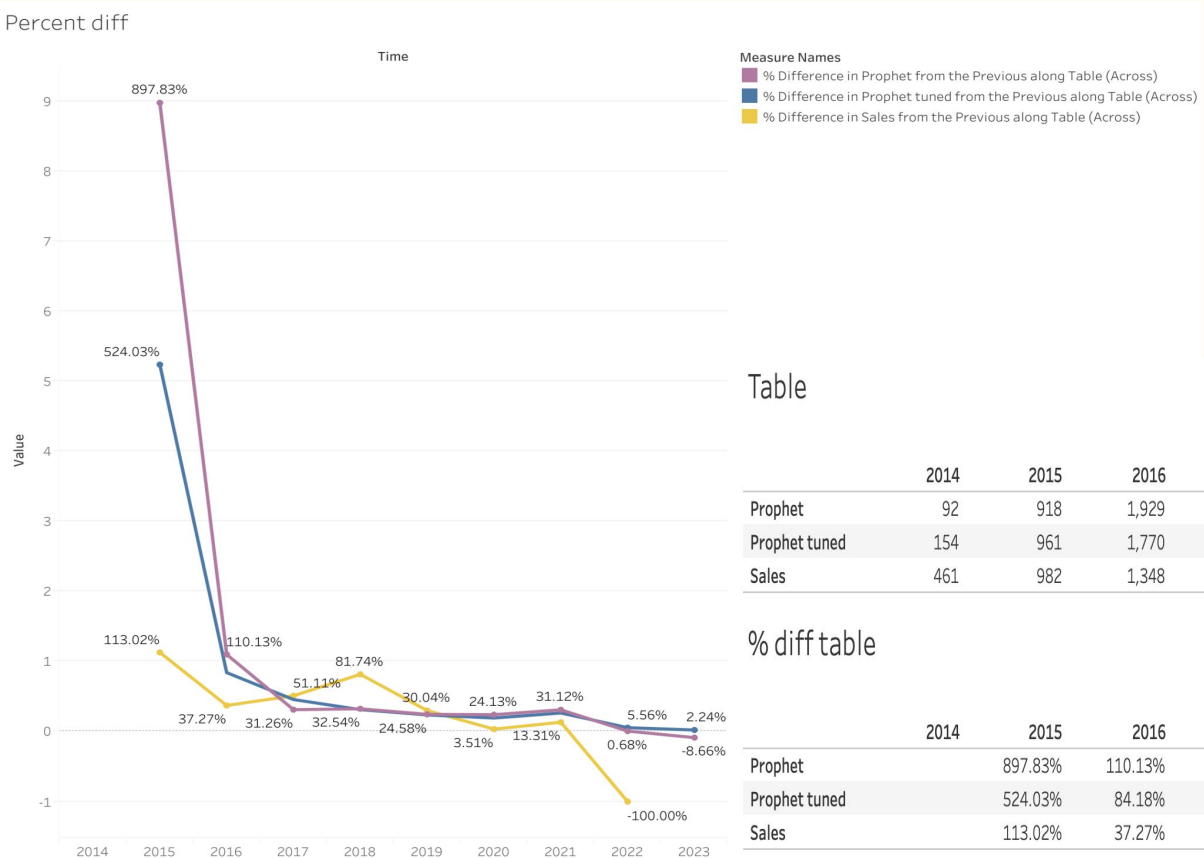
Comparisons of Models

Comparisons of two different forecasting models.





Percent difference over Years



This SKU has shown drastic growth pre-COVID then seems to level out (likely due to limited inventory in recent years from supply chain shortages).

Table

	Time									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Prophet	92	918	1,929	2,532	3,356	4,181	5,190	6,805	6,851	6,258
Prophet tuned	154	961	1,770	2,579	3,387	4,192	5,003	6,331	6,683	6,833
Sales	461	982	1,348	2,037	3,702	4,814	4,983	5,646		

% diff table

	Time									
	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Prophet		897.83%	110.13%	31.26%	32.54%	24.58%	24.13%	31.12%	0.68%	-8.66%
Prophet tuned		524.03%	84.18%	45.71%	31.33%	23.77%	19.35%	26.54%	5.56%	2.24%
Sales		113.02%	37.27%	51.11%	81.74%	30.04%	3.51%	13.31%	-100.00%	

Tableau Visuals:

https://public.tableau.com/app/profile/jae.park/viz/CovidforecastingwithProphet/Dashboard1

GitHub Repository:

https://github.com/2Delta/Project-4_MachineLearningIntegration/

Database Server Connection (Public Read):

jdbc:postgresql://database-1.cc8swew422eu.us-east-1.rds.amazonaws.com:5432/postgres

The End

(Literally)