ISE 535 – FINAL PROJECT- FALL 2020
Prediction of Product Demands in the Event of a Pandemic-like Scenario

Ashwin Ramachandran
ID: 200365498
aramach5@ncsu.edu

Introduction

- COVID-19 pandemic has resulted many industries to face major disruptions in their normal operations leading to major transitions to virtual learning, work-from-home, etc.

- The transitions have majorly affected demand across Computing products.

- As a result, most tech manufacturers are faced with taking crucial production decisions that impact the sales of their various product lines.

Goals and Objectives

- Assess influence of COVID on a state-level across all individual states.
- Identify influencing factors of Lenovo's product demand with regard to existing COVID-19 executive orders and study their contribution
- Build a predictive model to determine Lenovo's product demand in a future pandemic.
- Design and Build an interactive interface in Python that can be used by a person with limited data analysis experience

Workflow and Tools used

Data Cleaning

- Data Received had multiple datatypes and formats.
- Pandas and Jupyter Notebook used to initially sort data and add attributes available on different other sheets
- Pandas used to remove NaN values and convert datetime formats uniformly.
- Jupyter Notebooks used to visualize data while cleaning.

Data Analysis

- To observe trends and cycles in the data, pandas, matplotlib and Plotly.express were used.
- By describing dataframes and plotting relevant graphs, cycles were identified and variables were defined

Data Model Building

- Once Variables were defined, SciKit Learn's Ensemble tools were used to create various data models and validate them
- Final models were arrived based on Accuracy of prediction and Speed of execution

Storing Data model

- The **joblib** librairy was used to store the data models for re-use and the **json** library was used to store values that needed to be called for accuracy factor calculations

Interface Building

- The interface is built on Plotly Dash.

Data Cleaning Procedure

1. Matching Zip Codes with counties and cities

Import all Required Raw Data from csv Into Pandas DataFrame → Convert the Zip Codes in sales data from string to numerical entries → Create a dictionary of lists
Key: Zip code
Values: State, City, County
Save as a *.json file → Write function to read *.json and lookup values. Iterate over rows and call lookup function each time and write the result into new columns in sales DataFrame

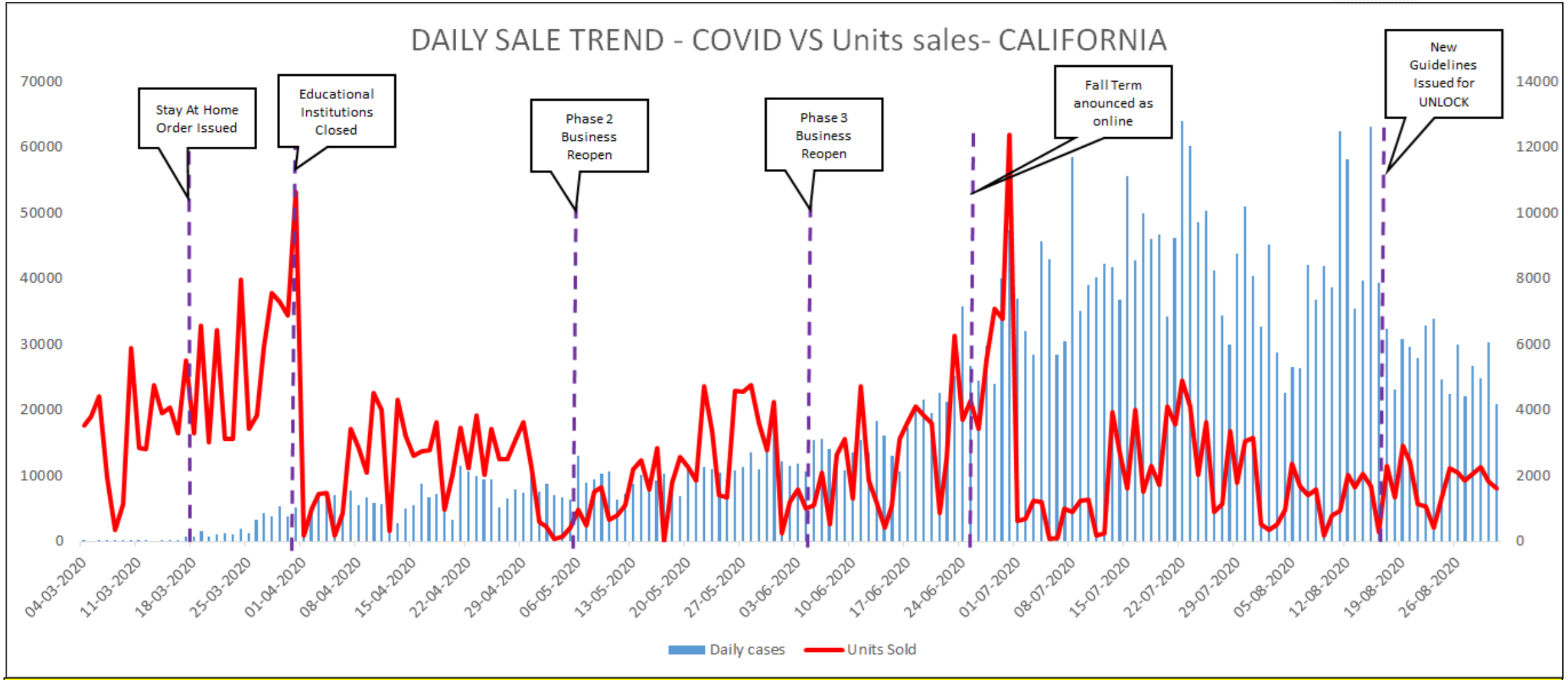
2. Matching Daily COVID data with counties and cities

Import all Required Raw Data from csv Into Pandas DataFrame → Convert the Dates in sales data from string to datetime entries → Create a dictionary of dictionaries for COVID
Key1: State, Key2: County, Key3: Date
Values: Cummulative COVID cases
Save as a *.json file → Write function to read *.json and lookup values. Iterate over rows and call lookup function each time and write the result into new column in sales DataFrame

3. Merge Data of all years

Use pd.concat to merge data from 2018 - 2020 → Use df.to_csv() to export and save cleaned file

Sales and COVID-19 Analysis



Methodology: Write plotting function using plotly.express to plot correlation graph as above to visualise sales w.r.t COVID evolution. (NOTE: Callouts given separately)

Defining Variables

All Possible variables were identified which would be readily available for input to the data model in case of another pandemic.

The Variables are split into 3 categories as below

- Median Age
- Education Level
- Unemployment Rate
- Mean Household Income
- Average Family Size
- Geographical Region
- Main Industry
- State
- County
- Race

Demographic

- Cumulative COVID-19 Cases
- State of Emergency
- Stay at Home Orders
- Quarantine Restrictions
- Face Mask Requirements
- School Closures
- Restaurant Restrictions
- Non-essential Business Restrictions
- Bar Restrictions

Pandemic Variables

- Brand
- Sub Brand
- Date of Sale
- Month of Sale
- Category (CCG)
- Class (LDW)

LENOVO

Data Model Building Procedure

Import all Required Raw Data from csv Into Pandas DataFrame

Convert the Dates in sales data from string to datetime entries

Use Sklearn Label Encoder Write a function to convert non-numerical entries to unique numpy labels

Use Train – Test – Split to generate test sets and training sets (75% train, 25% test used in this project)

Use Sklearn DecisionTrees and Ensemble Methods to create models

Using Xtest and Ytest, run cross validation on the results to check accuracy of prediction and optimum depths for Decision trees

Write function to create models based on each required parameter

Use feature importances to write importance of each variable to a Dataframe for each model

Eliminate non-essential variables and run another iteration

Choose DataModel that has best correlations of all important factors

In each loop, write Data model to a *.joblib file to make models persistent

Write function to call model from storage when required and make predictions

Model Selection

- Machine learning algorithms based on decision trees:
  - Decision Trees (Classifier / Regressor)
  - Random Forest (Classifier / Regressor)
  - Extra Trees (Classifier / Regressor)

- Rules to eliminate variables with low weightage:
  - Iteration 1 – Any with 0% Eliminated
  - Iteration 2 – Any with <5% Eliminated

Model Selection

Pandemic Based Model – Variable Weights and Accuracies

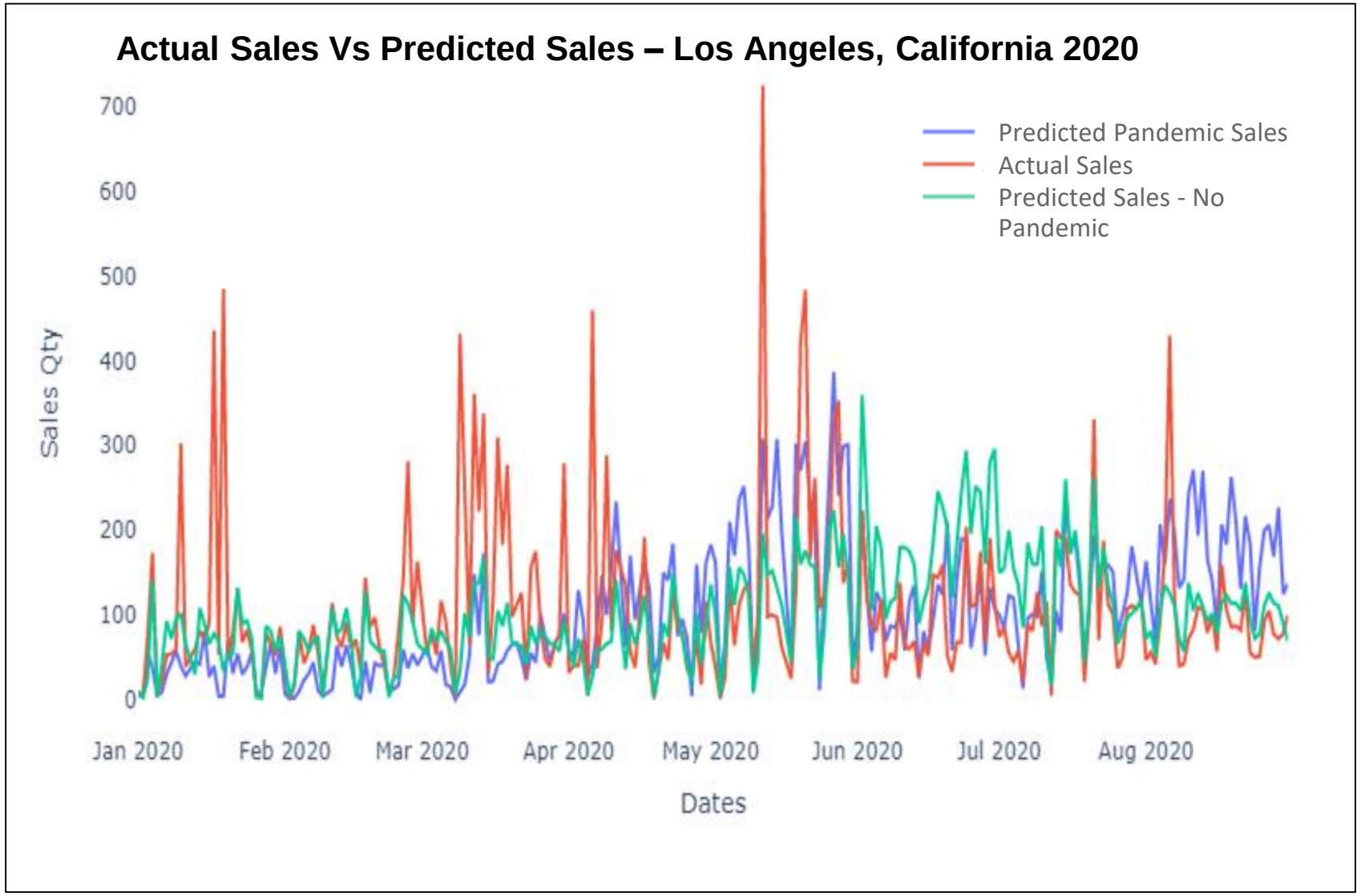
Features	RFC	RFR	ETC	ETR
State	2%	0%	3%	5%
County	13%	17%	16%	13%
Brand	3%	2%	4%	6%
Sub Brand	27%	20%	29%	18%
Bill Date	44%	47%	39%	40%
COVID cases	5%	5%	5%	6%
Months	5%	7%	3%	10%
Year	0%	1%	0%	3%
Sales Qty Accuracy	**95%**	**96%**	**88%**	**96%**

Best Model: The Extra Trees Regressor Algorithm

- Variables assigned optimal weights
- High Accuracy without overfitting
- Faster speed of execution without inhibiting accuracy - more efficient

- Random Forest Regressor: 874ms
- Extra Trees Regressor: 336ms

Model Selection



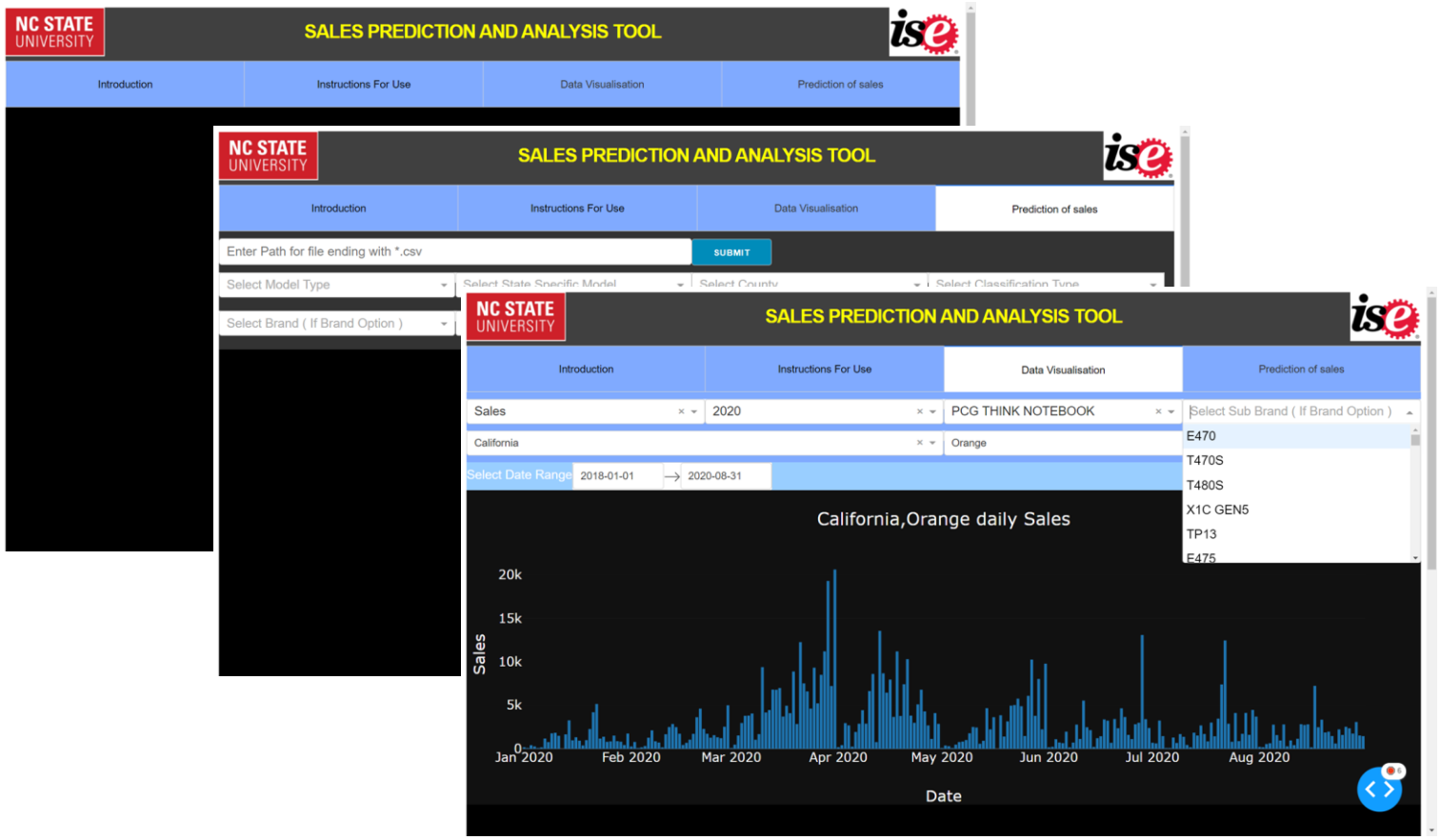
Prediction	Accuracy
Pandemic Model	96.1%
Non-Pandemic Model	96.4%

- Accuracy is defined as overall sales through the course of the Pandemic
- Model built with SciKit Learn Extra Trees Regressor Algorithm.

Building Sales Prediction Tool

App Server: http://127.0.0.1:8050/

- The final objective of the project was to build an interactive and easy to use tool that could be used by the layman with as few system requirements as possible
- Built on Python in Windows and using a Chrome based web browser as an in interface makes it as cross platform and open source as possible



Future Improvements to the Model

- The predictive model interface can be packaged into an application format
  - Currently runs on Python and a local server host created by Dash

Github Repository link: https://github.ncsu.edu/aramach5/Sales-Predictor-App

THANK YOU