sales_forecasting

May 4, 2025

1 Sales Forecasting Capstone Project

2 ——————————————-

3 1. Introduction

4 ——————————————-

5 This project aims to forecast item sales across restaurants using machine learning and deep learning.

6 We'll explore historical sales data, build predictive models, and compare performance.

[4]:
```
%brew install libomp

%pip install xgboost
```

```
==> Auto-updating Homebrew…
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man
brew`).
==> Downloading https://ghcr.io/v2/homebrew/portable-ruby/portable-

ruby/blobs/sha256:40e7f5d7514a7e9757facdd39006f7a351d3d7986d3a228be13c8b1c321672

7b
########################################################################## 100.0%
1.4%                                          25.8%          81.6%
==> Pouring portable-ruby-3.4.3.arm64_big_sur.bottle.tar.gz
==> Auto-updated Homebrew!
Updated 2 taps (homebrew/core and homebrew/cask).
==> New Formulae
ab-av1                                  leetgo
adaptivecpp                             libgoa
add-determinism                         libgudev
algolia                                 libpg_query
```

alloy
anubis
anyquery
apache-flink@1
api-linter
aqua
async_simple
bkmr
block-goose-cli
bombardier
bpmnlint
brename
buffrs
camlpdf
cargo-shear
cargo-sort
cargo-spellcheck
cf-terraforming
chdig
cloudfoundry-cli
code2prompt
cot
cpp-lazy
cspell
darklua
dbg-macro
dblab
decompose
dish
dockcheck
dockerfilegraph
dockerfmt
dtsroll
dvr-scan
dyff
easyeda2kicad
elfio
ente-cli
erlfmt
evans
excalidraw-converter
exomizer
fabric-ai
falcosecurity-libs
fancy-cat
fastly
fedify
ferron
libpostal-rest
lld@19
llvm@19
lume
mani
marmite
mcpm
md2pdf
mdq
mender-artifact
miniflux
mob
mox
multi-gitter
nak
netfetch
newsraft
nip4
nmstatectl
node-red
nom
nping
nuitka
opkssh
org-formation
otel-cli
otterdog
ov
pdfly
physunits
pivy
pkl-lsp
policy-engine
preevy
prog8
projectable
protobuf@29
protoc-gen-doc
punktf
pyp
ra-multiplex
rasterio
rattler-index
readerwriterqueue
restish
rhai
rofi
rpds-py

fiona
flow-control
fortitude
fortran-stdlib
foundry
fpm
fricas
ftxui
gama
gamdl
garnet
gcr
gdown
geesefs
gersemi
git-graph
git-who
github-mcp-server
globstar
gnome-online-accounts
goimports
gotz
gowall
grafana-alloy
gtree
gut
harsh
havener
hishtory
hjson
hk
hl
hub-tool
i686-elf-grub
iccmax
icu4c@77
iguana
immich-go
infat
infisical
intermodal
is-fast
jaguar
jenkins-cli
jira-cli
jsrepo
kafkactl
kapp
ruby-lsp
rustic
sacad
sdl3_ttf
semver
sentry-native
sequoia-chameleon-gnupg
sequoia-sqv
sexpect
shuttle
sitefetch
snowflake-cli
soft-serve
statesmith
sttr
sv2v
symfony-cli
taskflow
tdom
technitium-dns
technitium-library
terraform-iam-policy-validator
terraform-module-versions
tfcmt
tfmv
tfsort
tgpt
tmex
tml
todoist
trdsql
tsui
twitch-cli
typioca
uhubctl
umockdev
unciv
undercutf1
unoserver
veccore
vfkit
visidata
vscli
wfa2-lib
wrkflw
x-cmd
x86_64-elf-grub
xan

kbld
keyutils
kirimase
koji
krep
ktexttemplate
kubernetes-cli@1.32
largetifftools
lavat
lazyjournal
xk6
xtl
yalantinglibs
yoke
ytt
yutu
zlib-ng-compat
zlib-rs
zns

==> **New Casks**

ace-studio
advanced-renamer
agent-tars
antinote
atv-remote
automounterhelper
bambu-connect
batfi
beutl
block-goose
candy-crisis
captainplugins
charles@4
chime@alpha
chordpotion
cloudflare-warp@beta
cloudpouch
clover-chord-systems
comfyui
companion
companion-satellite
companion@beta
consul
coterm
dante-controller
dante-via
deepchat
desktime
earnapp
elemental
elemental@6
excire-foto
focu
font-adwaita
font-adwaita-mono-nerd-font
font-ancizar-sans
font-ancizar-serif
font-sf-mono-nerd-font-ligaturized
font-source-han-code-jp
font-special-gothic
font-special-gothic-condensed-one
font-special-gothic-expanded-one
font-wdxl-lubrifont-jp-n
font-wdxl-lubrifont-sc
font-wdxl-lubrifont-tc
font-webdings
font-winky-rough
font-winky-sans
fuse-t
gologin
granola
grayjay
hamrs-pro
highlight
hy-rpe2
ijhttp
inmusic-software-center
irpf2025
istatistica-core
jumpcloud-password-manager
k6-studio
kate
kilohearts-installer
liviable
luanti
macskk
meru
moment
mouseless@preview
ndi-tools
nethlink
notion-mail
nvidia-nsight-compute
obscura-vpn

```
font-aporetic
font-asta-sans
font-atkynson-mono-nerd-font
font-big-shoulders
font-big-shoulders-inline
font-big-shoulders-stencil
font-bizter
font-boldonse
font-bytesized
font-comic-relief
font-coral-pixels
font-epunda-sans
font-epunda-slab
font-exile
font-harmonyos-sans
font-harmonyos-sans-naskh-arabic
font-harmonyos-sans-sc
font-harmonyos-sans-tc
font-huninn
font-jetbrains-maple-mono
font-jetbrains-maple-mono-nf
font-kumar-one-outline
font-libertinus-math
font-libertinus-mono
font-lxgw-wenkai-gb-lite
font-m-plus-rounded-1c
font-maple-mono-normal
font-maple-mono-normal-cn
font-maple-mono-normal-nf
font-maple-mono-normal-nf-cn
font-noto-serif-dives-akuru
font-playpen-sans-arabic
font-playpen-sans-deva
font-playpen-sans-hebrew
font-playpen-sans-thai
font-pretendard-gov
opencloud
opera-air
outerbase-studio
ovice
pairpods
pareto-security
pastenow
pdl
precize
profit
qobuz-downloader
qt-design-studio
rave
realvnc-connect
repo-prompt
restapia
sc-menu
slidepad
sokim
soundanchor
stability-matrix
swift-shift
tal-drum
thelowtechguys-cling
trae
trae-cn
triliumnext-notes
ua-midi-control
veracrypt-fuse-t
vesktop
viables
vimy
voiceink
warp@preview
windsurf@next
yaak@beta

You have 12 outdated formulae installed.

==> Downloading

https://ghcr.io/v2/homebrew/core/libomp/manifests/20.1.4
########################################################################## 100.0%
==> Fetching libomp
==> Downloading

https://ghcr.io/v2/homebrew/core/libomp/blobs/sha256:d5a42b450ea
########################################################################## 100.0%
```

```
==> Pouring libomp--20.1.4.arm64_sonoma.bottle.tar.gz
==> Caveats
libomp is keg-only, which means it was not symlinked into /opt/homebrew,
because it can override GCC headers and result in broken builds.

For compilers to find libomp you may need to set:
  export LDFLAGS="-L/opt/homebrew/opt/libomp/lib"
  export CPPFLAGS="-I/opt/homebrew/opt/libomp/include"
==> Summary
   /opt/homebrew/Cellar/libomp/20.1.4: 9 files, 1.7MB
==> Running `brew cleanup libomp`…
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
==> `brew cleanup` has not been run in the last 30 days, running

now…
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
Removing: /Users/qunxu/Library/Caches/Homebrew/python@3.12--3.12.9… (15.6MB)
Removing: /Users/qunxu/Library/Logs/Homebrew/python@3.13… (2 files, 2KB)
```

[15]:

```python
# -------------------------------
# 2. Data Import & Preprocessing
# -------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

# Synthetic data loading placeholders
restaurants = pd.read_csv('resturants.csv')

items = pd.read_csv('items.csv')

sales = pd.read_csv('sales.csv')

# Dataset shapes
print(restaurants.shape, items.shape, sales.shape)

# Merge sales with items on item_id and id
sales = sales.merge(items, left_on='item_id', right_on='id', how='left')

# Merge the resulting sales with restaurants on store_id and id
sales = sales.merge(restaurants, left_on='store_id', right_on='id', how='left',␣
↪suffixes=('', '_store'))

# Select and rename columns for clarity
```

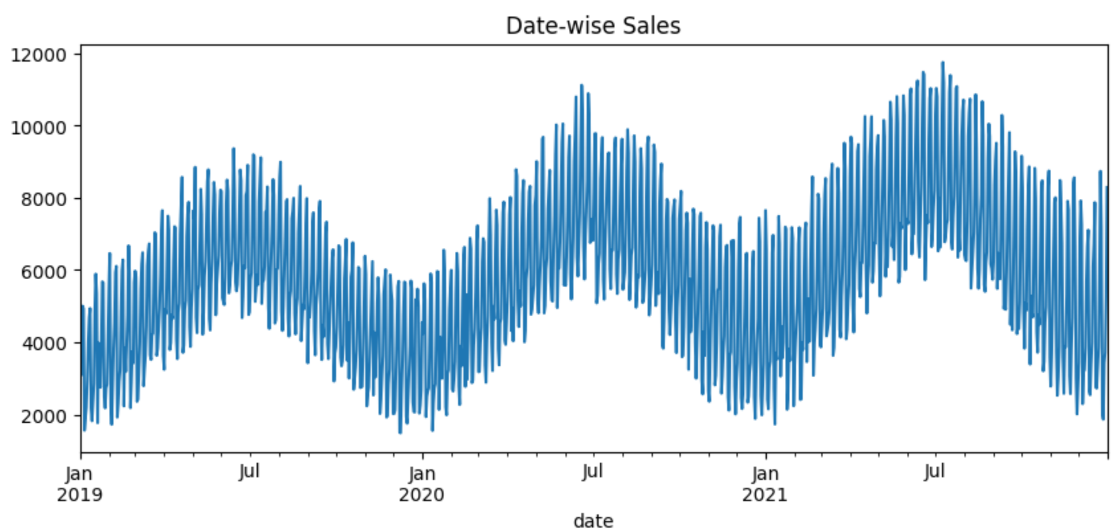
```
sales = sales[['date', 'store_id', 'name_store', 'item_id', 'name', 'price',␣
↪'item_count', 'kcal', 'cost']]
sales.columns = ['date', 'store_id', 'store_name', 'item_id', 'item_name',␣
↪'price', 'item_count', 'kcal', 'cost']

# Convert 'date' column to datetime format
sales['date'] = pd.to_datetime(sales['date'])

# Display the merged dataset
print(sales.head())
```

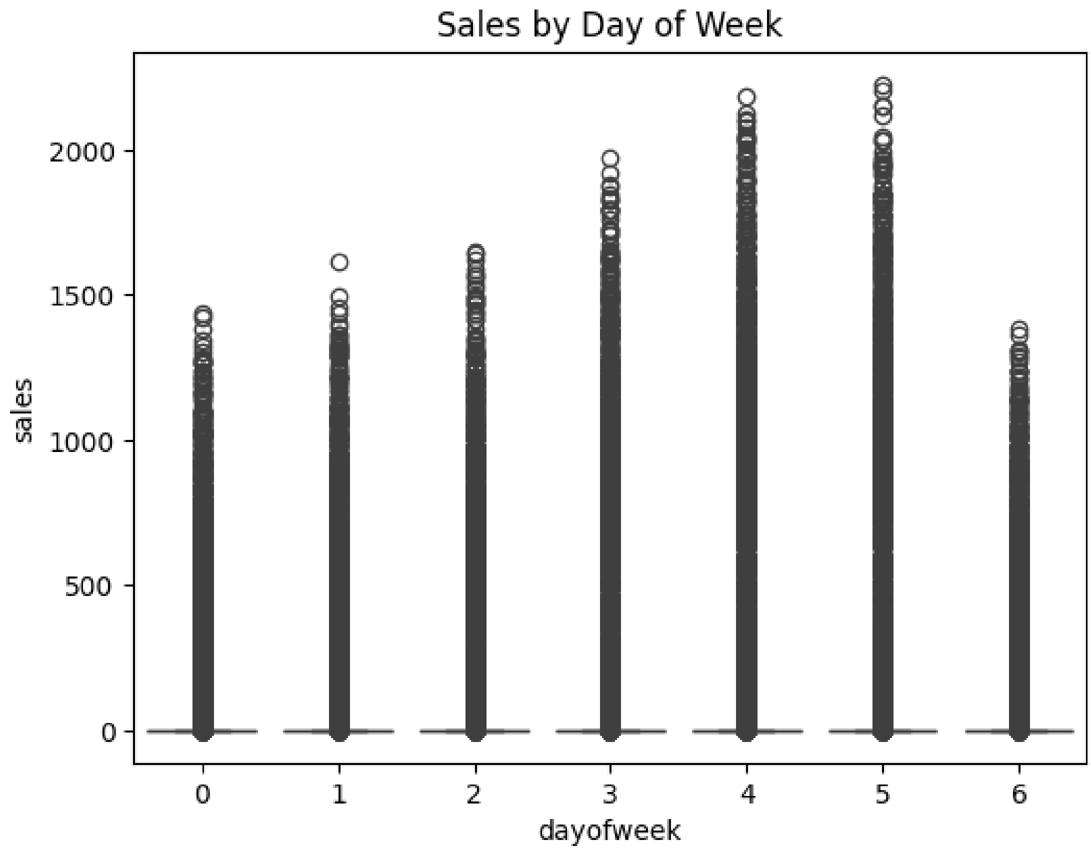
```
(6, 2) (100, 5) (109600, 4)
        date  store_id   store_name  item_id                        item_name  \
0 2019-01-01         1  Bob's Diner        3                Sweet Fruity Cake
1 2019-01-01         1  Bob's Diner        4  Amazing Steak Dinner with Rolls
2 2019-01-01         1  Bob's Diner       12             Fantastic Sweet Cola
3 2019-01-01         1  Bob's Diner       13          Sweet Frozen Soft Drink
4 2019-01-01         1  Bob's Diner       16             Frozen Milky Smoothy

   price  item_count  kcal   cost
0  29.22         2.0   931  29.22
1  26.42        22.0   763  26.42
2   4.87         7.0   478   4.87
3   4.18        12.0   490   4.18
4   3.21       136.0   284   3.21
```

[16]:

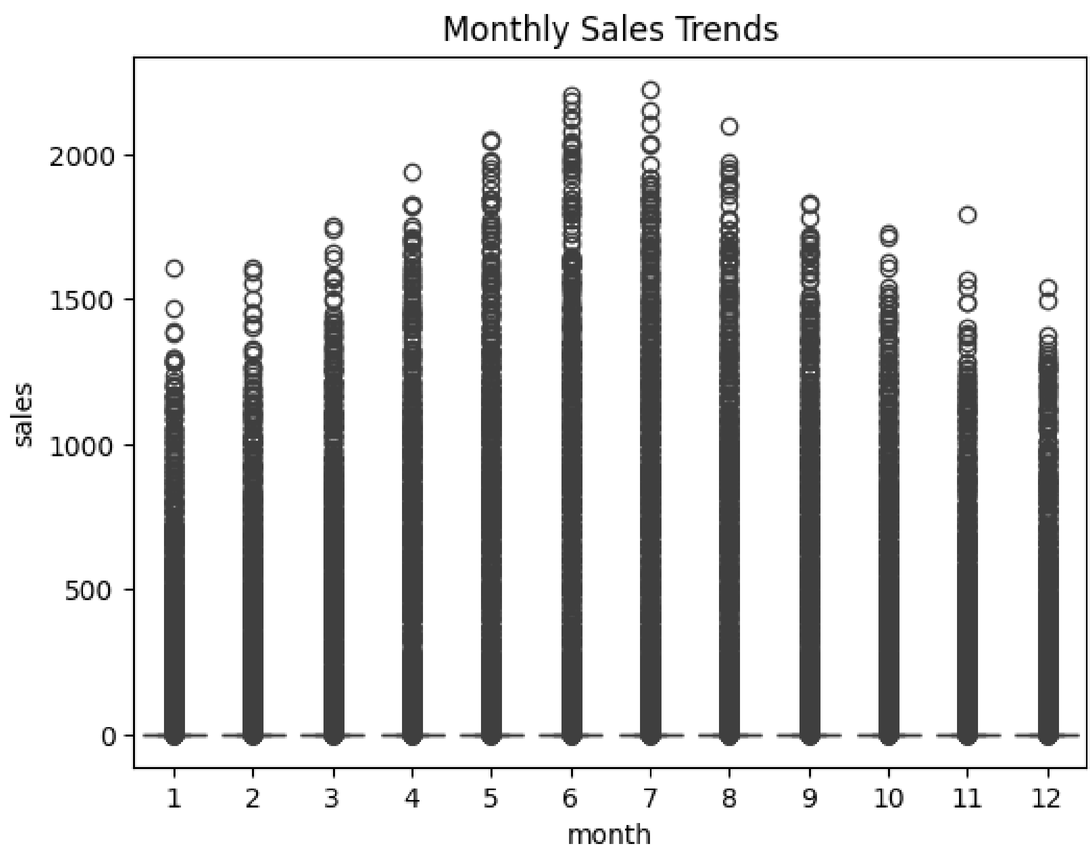
```
# -------------------------------
# 3. Exploratory Data Analysis
# -------------------------------
sales['sales'] = sales['item_count'] * sales['price']
sales['year'] = sales['date'].dt.year
sales['month'] = sales['date'].dt.month
sales['day'] = sales['date'].dt.day
sales['dayofweek'] = sales['date'].dt.dayofweek
sales['quarter'] = sales['date'].dt.quarter

# a. Date-wise sales
sales.groupby('date')['sales'].sum().plot(figsize=(10,4), title='Date-wise␣
↪Sales')
plt.show()

# b. Weekly pattern
sns.boxplot(x='dayofweek', y='sales', data=sales)
plt.title('Sales by Day of Week')
plt.show()

# c. Monthly trends
```

```
sns.boxplot(x='month', y='sales', data=sales)
plt.title('Monthly Sales Trends')
plt.show()

# d. Quarterly pattern
quarter_avg = sales.groupby(['year', 'quarter'])['sales'].mean().unstack()
quarter_avg.plot(kind='bar', title='Quarterly Sales Avg')
plt.show()

# e. Restaurant performance
restaurant_sales = sales.groupby('store_name')['sales'].sum().
 ↪sort_values(ascending=False)
restaurant_sales.plot(kind='bar', title='Total Sales per Restaurant')
plt.show()

# f. Most popular items
popular_items = sales.groupby('item_name')['item_count'].sum().
 ↪sort_values(ascending=False)
print("Popular items:\n", popular_items)

# g. Revenue vs volume
store_revenue = sales.groupby('store_name')['sales'].sum()
store_volume = sales.groupby('store_name')['item_count'].sum()
print(pd.DataFrame({'Revenue': store_revenue, 'Volume': store_volume}))

# h. Most expensive items per restaurant
expensive = sales.groupby(['store_name', 'item_name']).agg({'price': 'max',␣
 ↪'kcal': 'mean'})
print("Most expensive items with calories:\n", expensive)
```

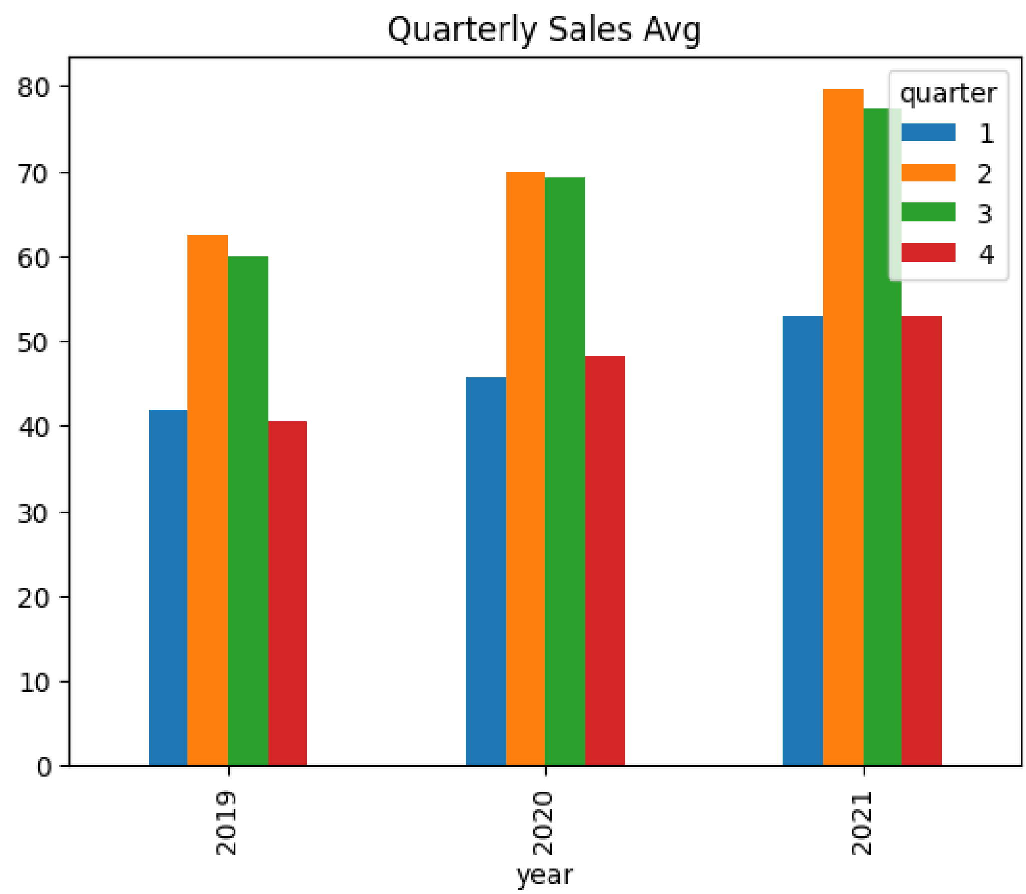


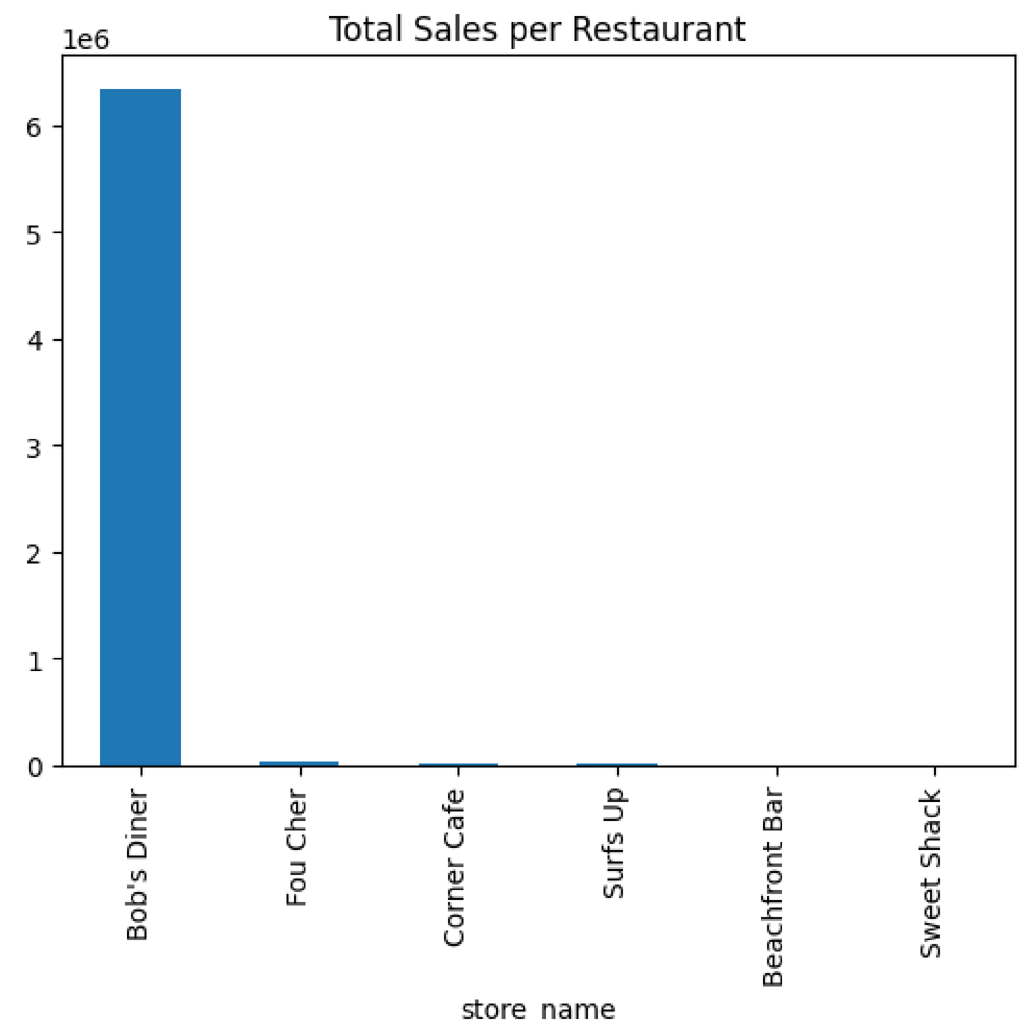


Sales by Day of Week
sales
dayofweek
2000
1500
1000
500
0
0
1
2
3
4
5
6



Monthly Sales Trends
sales
0
500
1000
1500
2000
1
2
3
4
5
6
7
8
9
10
11
12
month


Quarterly Sales Avg
quarter
1
2
3
4
80
70
60
50
40
30
20
10
0
2019
2020
2021
year




```
Popular items:
 item_name
Strawberry Smoothy                             236337.0
Frozen Milky Smoothy                           103263.0
Amazing pork lunch                              61043.0
Mutton Dinner                                   52772.0
Orange Juice                                    43874.0
                                                  …
Original Milky Cake                                 0.0
Fantastic Fruity Salmon with Bread meal             0.0
Awesome Fruity Lamb with Vegetables Dinner          0.0
Blue Ribbon Frozen Milky Cake                       0.0
Original Fruity Carrot Cake                         0.0
Name: item_count, Length: 94, dtype: float64
                   Revenue    Volume
```

```
store_name
Beachfront Bar     3796.20    1305.0
Bob's Diner     6337275.69  687527.0
Corner Cafe       16551.43    1310.0
Fou Cher          27885.37    1106.0
Surfs Up          15651.49    1803.0
Sweet Shack        2578.27    1736.0
Most expensive items with calories:
                                                  price   kcal
store_name     item_name
Beachfront Bar Awesome Vodka Cocktail              2.48  223.0
               Fantastic Milky Smoothy             2.91  318.0
               Original Crazy Cocktail             2.43  228.0
               Original Gin Cocktail               2.99  279.0
               Original Sweet Milky Soft Drink     5.00  579.0
…                                                 …     …
Sweet Shack    Fantastic Cake                      5.08  525.0
               Fantastic Milky Smoothy             5.11  383.0
               Milky vegi Smoothy                  3.86  150.0
               Original Milky Cake                 6.50  595.0
               Original Sweet Milky Soft Drink     5.68  535.0

[100 rows x 2 columns]
```

[17]:

```python
# -------------------------------
# 4. Forecasting with ML Models
# -------------------------------
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error

# Feature engineering
features = sales[['date', 'sales']].copy()
features['day'] = features['date'].dt.day
features['month'] = features['date'].dt.month
features['year'] = features['date'].dt.year
features['dayofweek'] = features['date'].dt.dayofweek
features['quarter'] = features['date'].dt.quarter

X = features[['day', 'month', 'year', 'dayofweek', 'quarter']]
y = features['sales']

# Train-test split (last 6 months)
split_date = features['date'].max() - pd.DateOffset(months=6)
train = features[features['date'] <= split_date]
```

```
test = features[features['date'] > split_date]

X_train = train[['day', 'month', 'year', 'dayofweek', 'quarter']]
y_train = train['sales']
X_test = test[['day', 'month', 'year', 'dayofweek', 'quarter']]
y_test = test['sales']

# Models
models = {
    'LinearRegression': LinearRegression(),
    'RandomForest': RandomForestRegressor(),
    'XGBoost': XGBRegressor()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, preds))
    print(f'{name} RMSE: {rmse:.2f}')
```

```
LinearRegression RMSE: 237.33
RandomForest RMSE: 236.23
XGBoost RMSE: 236.24
```

[18]:
```
# -------------------------------
# 5. Forecasting with LSTM
# -------------------------------
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Use total sales per day
daily_sales = sales.groupby('date')['sales'].sum().reset_index()
scaler = MinMaxScaler()
daily_sales['scaled_sales'] = scaler.fit_transform(daily_sales[['sales']])

# Create sequence data
def create_sequence(data, seq_len):
    X, y = [], []
    for i in range(len(data) - seq_len):
        X.append(data[i:i+seq_len])
        y.append(data[i+seq_len])
    return np.array(X), np.array(y)

sequence_length = 10
X_seq, y_seq = create_sequence(daily_sales['scaled_sales'].values,
 ↪sequence_length)
```

```python
# Train/test split
split = int(len(X_seq) * 0.8)
X_train_lstm, X_test_lstm = X_seq[:split], X_seq[split:]
y_train_lstm, y_test_lstm = y_seq[:split], y_seq[split:]

# LSTM model
model_lstm = Sequential([
    LSTM(50, activation='relu', input_shape=(sequence_length, 1)),
    Dense(1)
])
model_lstm.compile(optimizer='adam', loss='mse')
model_lstm.fit(X_train_lstm[..., np.newaxis], y_train_lstm, epochs=10,␣
 ↪verbose=1)

# Predict and evaluate
preds_lstm = model_lstm.predict(X_test_lstm[..., np.newaxis])
mape = np.mean(np.abs((y_test_lstm - preds_lstm.flatten()) / y_test_lstm)) * 100
print(f"LSTM MAPE: {mape:.2f}%")
```

```
Epoch 1/10
```

```
/Users/qunxu/Documents/guild/AI/guild_projects/.venv/lib/python3.12/site-
packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
28/28           1s 2ms/step - loss:
0.1788
Epoch 2/10
28/28           0s 1ms/step - loss:
0.0471
Epoch 3/10
28/28           0s 1ms/step - loss:
0.0294
Epoch 4/10
28/28           0s 1ms/step - loss:
0.0284
Epoch 5/10
28/28           0s 2ms/step - loss:
0.0269
Epoch 6/10
28/28           0s 1ms/step - loss:
0.0263
Epoch 7/10
28/28           0s 2ms/step - loss:
0.0233
```

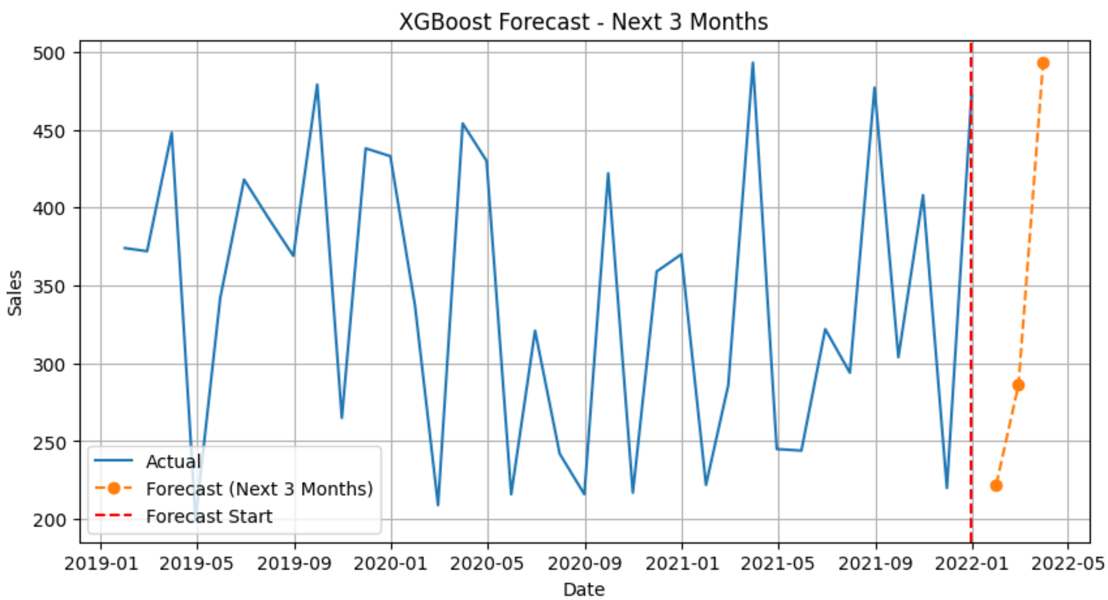
```
Epoch 8/10
28/28              0s 1ms/step - loss:
0.0212
Epoch 9/10
28/28              0s 1ms/step - loss:
0.0182
Epoch 10/10
28/28              0s 1ms/step - loss:
0.0147
7/7            0s 9ms/step
LSTM MAPE: 36.94%
```

[25]:

```python
# Forecast Next 3 Months with XGBoost

# Simulated data - replace with your actual dataset
date_range = pd.date_range(start="2019-01-01", periods=36, freq='M')
sales = np.random.randint(200, 500, size=len(date_range))
df = pd.DataFrame({"date": date_range, "sales": sales})

# Feature engineering
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year

X = df[['month', 'year']]
y = df['sales']

# Train on full dataset
xgb_model = XGBRegressor()
xgb_model.fit(X, y)

# Forecast next 3 months
last_date = df['date'].max()
future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),
 ↪periods=3, freq='M')
future_df = pd.DataFrame({'date': future_dates})
future_df['month'] = future_df['date'].dt.month
future_df['year'] = future_df['date'].dt.year

# Predict
future_df['forecast'] = xgb_model.predict(future_df[['month', 'year']])

# Plot actual + forecast
plt.figure(figsize=(10, 5))
plt.plot(df['date'], df['sales'], label="Actual")
plt.plot(future_df['date'], future_df['forecast'], label="Forecast (Next 3
 ↪Months)", linestyle='--', marker='o')
plt.axvline(x=last_date, color='red', linestyle='--', label="Forecast Start")
```

```
plt.title("XGBoost Forecast - Next 3 Months")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.legend()
plt.grid()
plt.show()
```

```
/var/folders/9l/kd0gvnp51998_xw_1lzf4nf40000gn/T/ipykernel_5660/3815182634.py:4:
FutureWarning: 'M' is deprecated and will be removed in a future version, please
use 'ME' instead.
  date_range = pd.date_range(start="2019-01-01", periods=36, freq='M')
/var/folders/9l/kd0gvnp51998_xw_1lzf4nf40000gn/T/ipykernel_5660/3815182634.py:21
: FutureWarning: 'M' is deprecated and will be removed in a future version,
please use 'ME' instead.
  future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),
periods=3, freq='M')
```



[27]:
```
# Forecast Next 3 Months with LSTM

# Simulated data - replace with your actual dataset
date_range = pd.date_range(start="2019-01-01", periods=36, freq='M')
sales = np.random.randint(200, 500, size=len(date_range))
df = pd.DataFrame({"date": date_range, "sales": sales})
df.set_index('date', inplace=True)

# Normalize
scaler = MinMaxScaler()
```

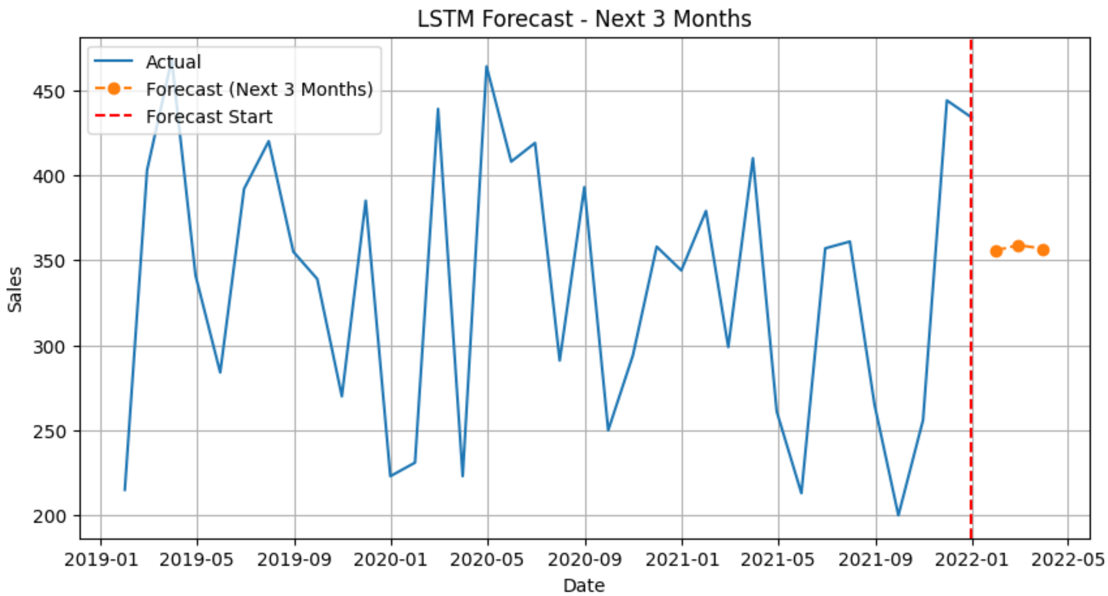
```python
scaled_data = scaler.fit_transform(df[['sales']])

# Prepare LSTM input
def create_sequences(data, look_back=3):
    X, y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:i+look_back])
        y.append(data[i+look_back])
    return np.array(X), np.array(y)

look_back = 3
X, y = create_sequences(scaled_data, look_back)
X = X.reshape((X.shape[0], X.shape[1], 1))

# LSTM Model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)

# Forecast next 3 steps
last_seq = scaled_data[-look_back:].reshape((1, look_back, 1))
predictions = []
for _ in range(3):
    next_pred = model.predict(last_seq)[0]
    predictions.append(next_pred)
    last_seq = np.append(last_seq[:, 1:, :], next_pred.reshape(1, 1, 1), axis=1)

# Inverse scale predictions
predicted_sales = scaler.inverse_transform(predictions)

# Build forecast dates
last_date = df.index.max()
future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),␣
 ↪periods=3, freq='M')

# Plot
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['sales'], label='Actual')
plt.plot(future_dates, predicted_sales.flatten(), label='Forecast (Next 3␣
 ↪Months)', linestyle='--', marker='o')
plt.axvline(x=last_date, color='red', linestyle='--', label="Forecast Start")
plt.title("LSTM Forecast - Next 3 Months")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.legend()
```

```
plt.grid()
plt.show()
```

```
/var/folders/9l/kd0gvnp51998_xw_1lzf4nf40000gn/T/ipykernel_5660/2943951659.py:4:
FutureWarning: 'M' is deprecated and will be removed in a future version, please
use 'ME' instead.
  date_range = pd.date_range(start="2019-01-01", periods=36, freq='M')
/Users/qunxu/Documents/guild/AI/guild_projects/.venv/lib/python3.12/site-
packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
WARNING:tensorflow:5 out of the last 10 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x16972ff60> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/1            0s 60ms/step
1/1            0s 15ms/step
1/1            0s 12ms/step
```

```
/var/folders/9l/kd0gvnp51998_xw_1lzf4nf40000gn/T/ipykernel_5660/2943951659.py:45
: FutureWarning: 'M' is deprecated and will be removed in a future version,
please use 'ME' instead.
  future_dates = pd.date_range(start=last_date + pd.offsets.MonthBegin(1),
periods=3, freq='M')
```


LSTM Forecast - Next 3 Months
Actual
Forecast (Next 3 Months)
Forecast Start
Sales
Date
450
400
350
300
250
200
2019-01
2019-05
2019-09
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05