Time-Series Forecasting Homework

Answer the Following Questions in the Jupyter Notebook

If you have any questions, reach out to bobby@prizepicks.com

Feel free to use any packages you want to fit, display, analyze, etc..

Patrick Hayes - 7/27/2023

Instructions

using the forecast_train.csv - fit two time-series forecasting models to the daily data, one for market A and one for market B

Sounds good, let's go!

In [ ]:
```
## IMPORTS ##

import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tools.sm_exceptions import ValueWarning
from pmdarima import auto_arima

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller, acf

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, mean_absolute_percentage_error
import matplotlib.pyplot as plt
from datetime import datetime
from pandas.plotting import autocorrelation_plot
import warnings

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
warnings.simplefilter("ignore", ValueWarning)

#############
```

Light EDA and Data Cleaning

My intentions here are to get a general understanding of how each market. I looked for general shapes, patterns, inconsistencies, etc. Some were removed as I proceeded, but I left some in for future me/you.

With more time, I would have cleaned up and turned the import process into functions.

In [ ]:
```
forecast_train_df = pd.read_csv('forecast_train.csv')
forecast_test_df = pd.read_csv('forecast_test.csv')
```

In [ ]:
```
forecast_test_df.head()
```

Out[ ]:

	market_name	date	target_kpi
0	Market A	2021-12-17	151852
1	Market A	2021-12-18	95325
2	Market A	2021-12-19	71607
3	Market A	2021-12-20	85948
4	Market A	2021-12-21	101253

In [ ]:
```
forecast_train_df.shape, forecast_test_df.shape
```

Out[ ]:
```
((120, 3), (30, 3))
```

In [ ]:
```
forecast_train_df.market_name.value_counts()
```

Out[ ]:
```
market_name
Market A    60
Market B    60
Name: count, dtype: int64
```

In [ ]:
```
forecast_test_df.market_name.value_counts()
```

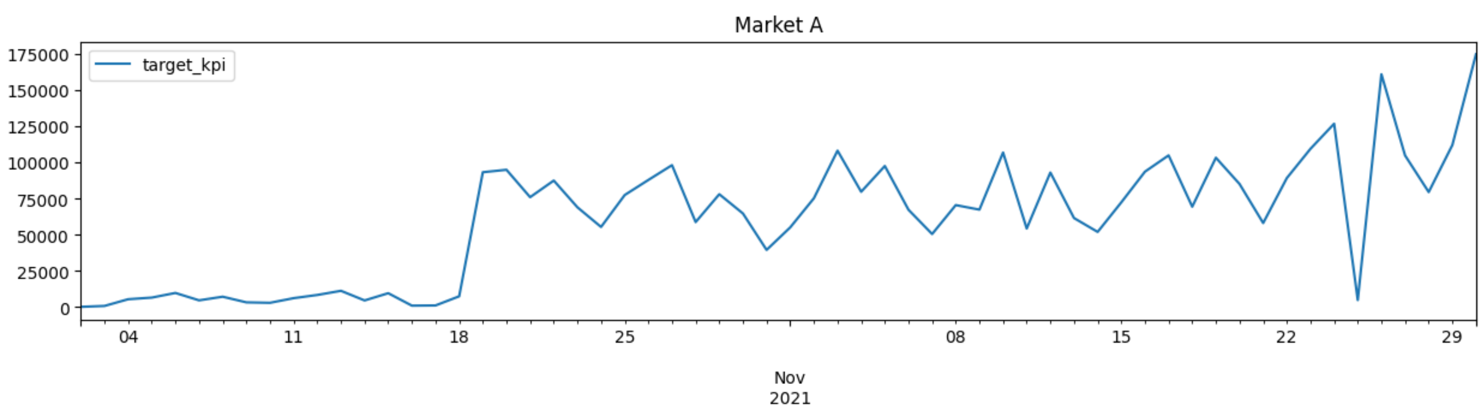
Out[ ]:
```
market_name
Market A    15
Market B    15
Name: count, dtype: int64
```

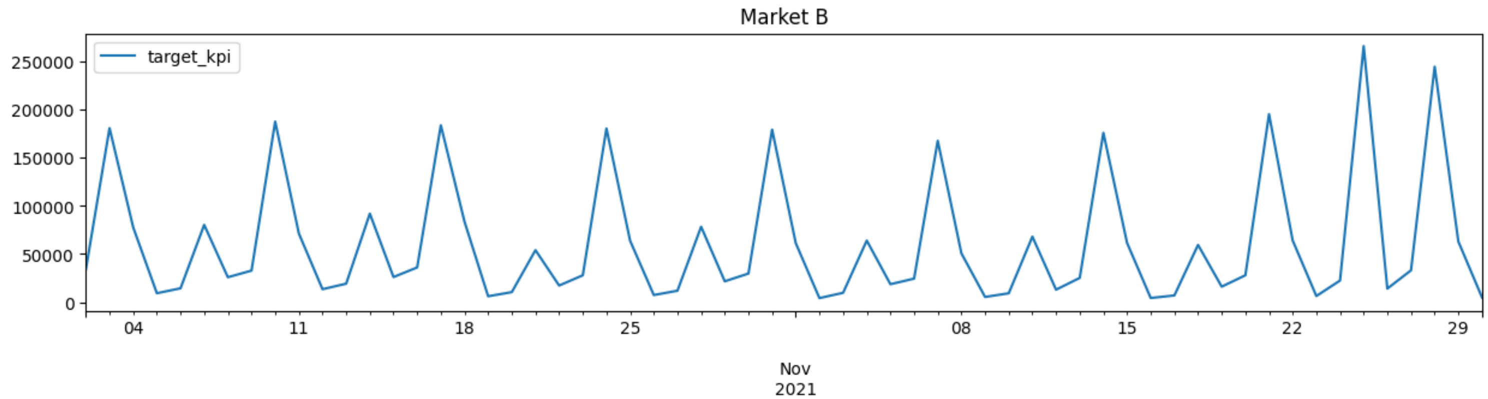
In [ ]:
```
#serparting train/test for each market, function would be used in future

ma_train_df = forecast_train_df[forecast_train_df['market_name'] == 'Market A']
ma_test_df = forecast_test_df[forecast_test_df['market_name'] == 'Market A']
mb_train_df = forecast_train_df[forecast_train_df['market_name'] == 'Market B']
mb_test_df = forecast_test_df[forecast_test_df['market_name'] == 'Market B']
```

In [ ]:
```
#swift clean up for each
def clean_df(df):
    df = df.drop(columns=['market_name'])
    df['date'] = pd.to_datetime(df['date'])
    df = df.set_index('date')
    df.index.name = None
    return df
```

In [ ]:
```
#repetitive work that again would be better suited from a function with more time to work
ma_train_df = clean_df(ma_train_df)
ma_test_df = clean_df(ma_test_df)
mb_train_df = clean_df(mb_train_df)
mb_test_df = clean_df(mb_test_df)
```

In [ ]:
```
#show me what you're woring with, markets!
#Market A, what could you be?
#Market B, I can't wait for football either.

ma_train_df.plot(title= 'Market A', figsize=(15, 3))
mb_train_df.plot(title= 'Market B', figsize=(15, 3))
plt.show()
```





ARIMA Model parameter exploration and selection

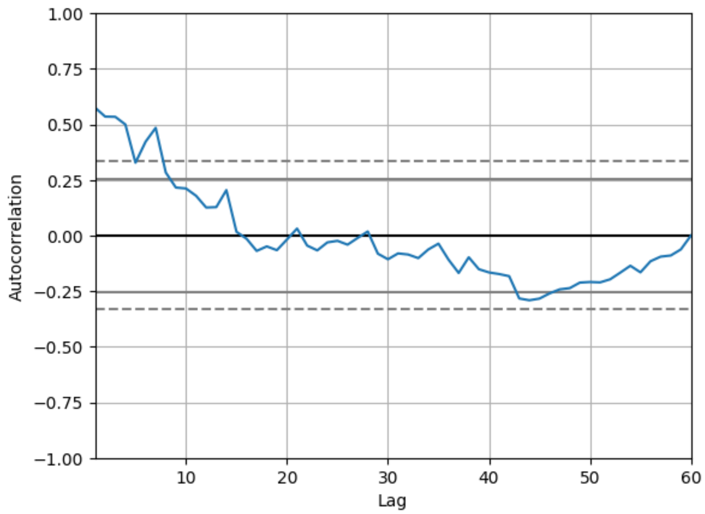
In [ ]:
```
### Statrionary test

#We like stationary models as it suggests mean, variance and autocorrelation are constant over time.

#both markets as imported are not stationary, which means we'll need to flip the "d" in ARIMA to 1 or 2.
def is_stationary(df):
    result = adfuller(df)
    return result[1] <= 0.05


market_a_stationary = is_stationary(ma_train_df)
market_b_stationary = is_stationary(mb_train_df)
market_a_stationary, market_b_stationary
```

Out[ ]:
```
(False, False)
```

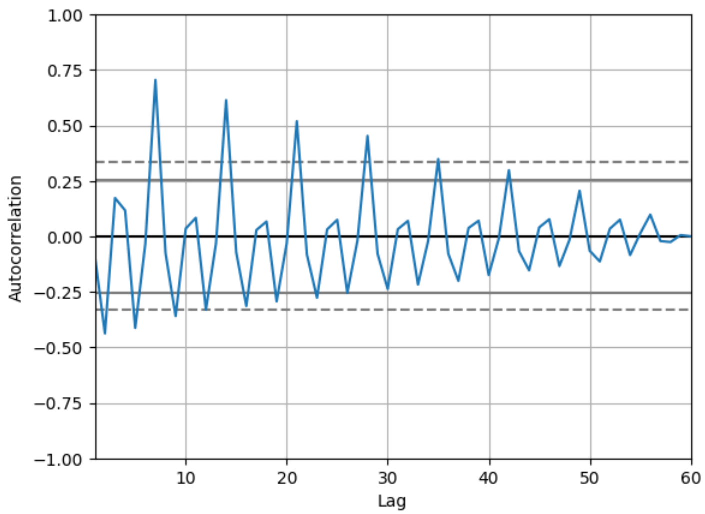
In [ ]:
```
#confirming here that by using diffs, the markets are now stationary
ma_train_df_diff = ma_train_df.diff().dropna()
mb_train_df_diff = mb_train_df.diff().dropna()
is_stationary(ma_train_df_diff), is_stationary(mb_train_df_diff)
```

Out[ ]:
```
(True, True)
```

In [ ]:
```
# Autocorrelation to determine p for Market A
# I know sports typically revolve around weekly schedules, and there's a noticeable
# drop off after the 7th lag. 7 will be used as the "p" value for Market A.

autocorrelation_plot(ma_train_df)
plt.show()
```



In [ ]:
```
# Autocorrelation to determine p for Market B

# Market B a little more complex. There's a large spike at 5, but there is a repeating pattern
# this one I'll explore with the ACF and PACF plots, plus the auto_arima function.

autocorrelation_plot(mb_train_df)
plt.show()
```



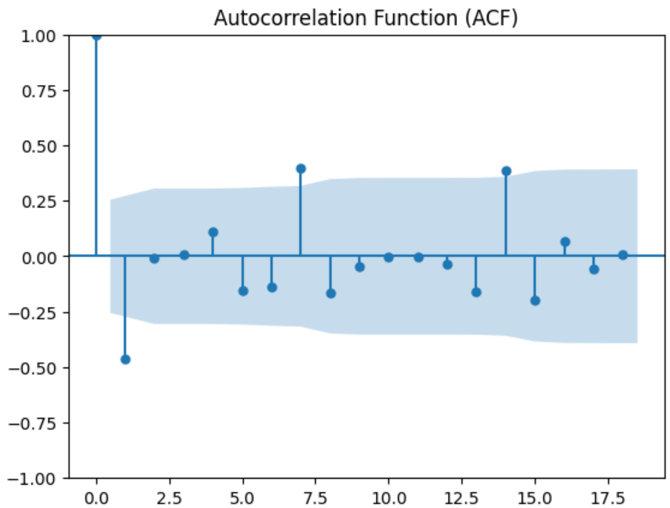
ACF and PACF plots

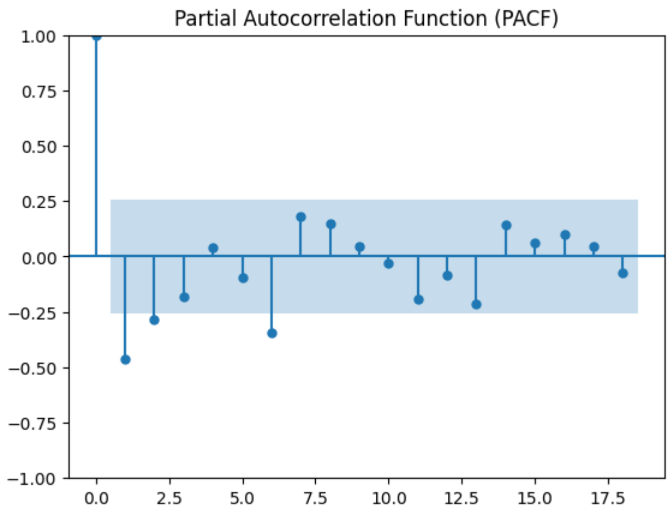
Most versions commented out to save space, but admittedly an area of review that I want to improve in.

In [ ]:
```
# plot_acf(ma_train_df)
# plt.title("Autocorrelation Function (ACF)")
# plt.show()
```

```
# plot_pacf(ma_train_df)
# plt.title("Partial Autocorrelation Function (PACF)")
# plt.show()
```

In [ ]:
```
# using the diff dataframes for Market A, 7 jumps out.

plot_acf(ma_train_df_diff)
plt.title("Autocorrelation Function (ACF)")
plt.show()

plot_pacf(ma_train_df_diff)
plt.title("Partial Autocorrelation Function (PACF)")
plt.show()
```





In [ ]:
```
# plot_acf(mb_train_df, lags=7)
# plt.title("Autocorrelation Function (ACF)")
# plt.show()
```

```
# plot_pacf(mb_train_df, lags=7)
# plt.title("Partial Autocorrelation Function (PACF)")
# plt.show()

# plot_acf(mb_train_df_diff)
# plt.title("Autocorrelation Function (ACF)")
# plt.show()

# plot_pacf(mb_train_df_diff)
# plt.title("Partial Autocorrelation Function (PACF)")
# plt.show()
```

In [ ]:
```
# Was curious for the auto output with model B due to it's repeating pattern of autocorrelation

# it suggets using (1,0,3), and I attempted it, but the performance was awful and I aborted.

ab = auto_arima(mb_train_df, stepwise=False, seasonal=False)
ab.summary()
```

Out[ ]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	60
Model:	SARIMAX(1, 0, 3)	**Log Likelihood**	-746.826
Date:	Thu, 27 Jul 2023	**AIC**	1503.652
Time:	18:54:14	**BIC**	1514.124
Sample:	10-02-2021	**HQIC**	1507.748
	- 11-30-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9882	0.028	35.479	0.000	0.934	1.043
ma.L1	-1.0796	0.198	-5.441	0.000	-1.469	-0.691
ma.L2	-0.4121	0.206	-1.996	0.046	-0.817	-0.007
ma.L3	0.6897	0.213	3.232	0.001	0.271	1.108
sigma2	4.816e+09	3.47e-11	1.39e+20	0.000	4.82e+09	4.82e+09

Ljung-Box (L1) (Q):	0.32	**Jarque-Bera (JB):**	14.98
Prob(Q):	0.57	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.22	**Skew:**	1.07
Prob(H) (two-sided):	0.08	**Kurtosis:**	4.20

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.64e+35. Standard errors may be unstable.

Model Fitting

I started with a simple ARIMA model. I pulled the p and d values from above and experimented with q values to find the best-performing one. I initially thought I'd be using different parameters for each model, but they ended up being the same.

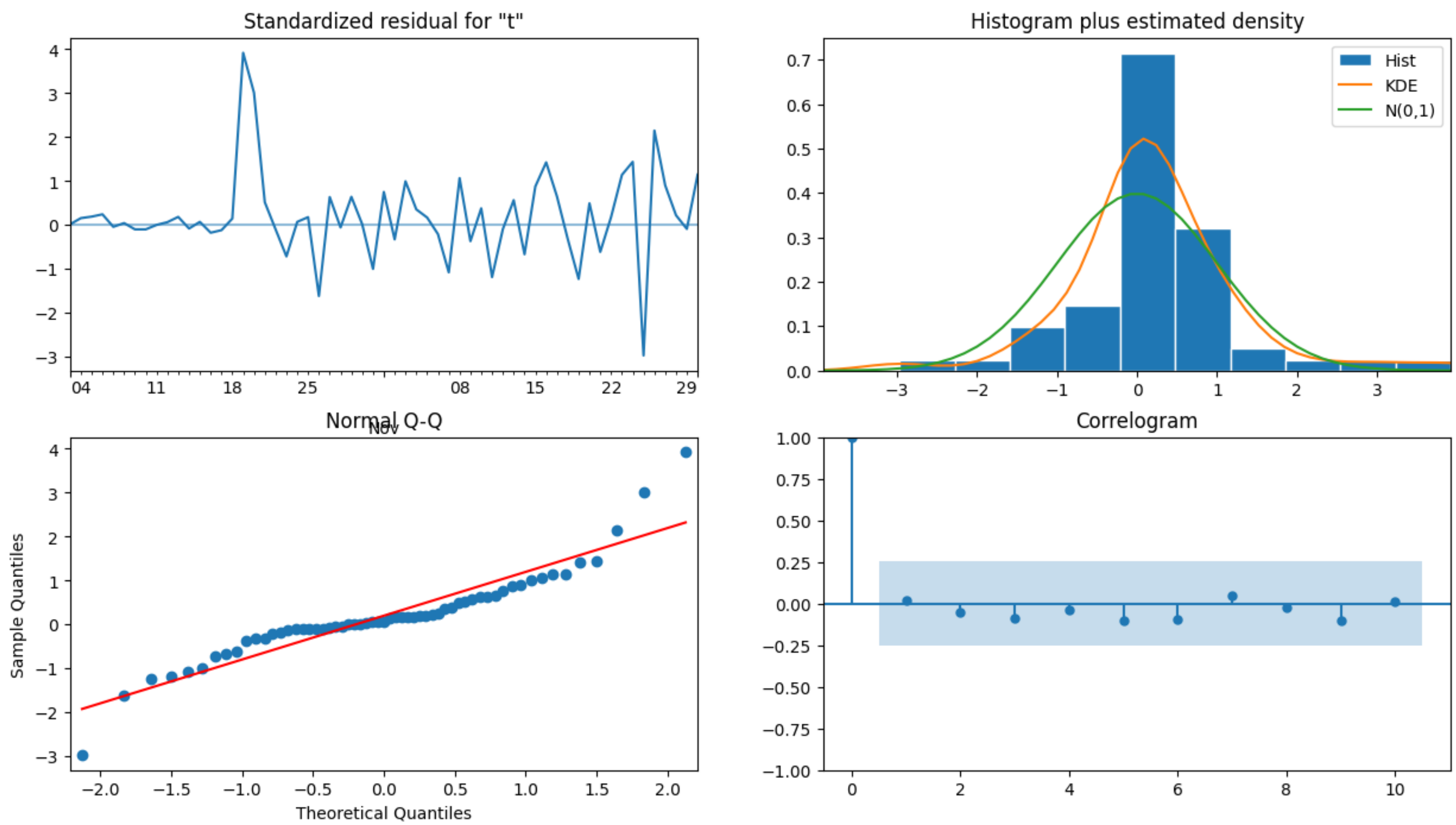
In [ ]:
```
### MARKET A MODEL ###

p, d, q = 7, 1, 7  # autoregression, differencing, moving average
ma_model = ARIMA(ma_train_df, order=(p, d, q))
ma_model_fit = ma_model.fit()
```

```
ma_predict = ma_model_fit.predict(start=ma_train_df.index[0], end=ma_train_df.index[-1])

mse = round(mean_squared_error(ma_train_df, ma_predict),2)
ma_rmse = round(np.sqrt(mse),2)
mae = round(mean_absolute_error(ma_train_df, ma_predict),2)
r2 = round(r2_score(mb_train_df, mb_predict),2)

plt.figure(figsize=(10, 4))
plt.plot(ma_train_df.index, ma_train_df, label='Train Data')
plt.plot(ma_predict.index, ma_predict, label='Model Predictions')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title("Market A - Model Fit to Train Data\n" + f'RMSE: {ma_rmse} MAE: {mae} R2: {r2}')
plt.legend()
plt.show()
```

```
/Users/patrick/Documents/Sports Analytics/prizepicks/venv/lib/python3.11/site-packages/statsmodels/tsa/statespace/sarimax.py:966: UserWarning: Non-stationary starting autoregressive paramet
ers found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
/Users/patrick/Documents/Sports Analytics/prizepicks/venv/lib/python3.11/site-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge.
Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[19], line 12
     10 ma_rmse = round(np.sqrt(mse),2)
     11 mae = round(mean_absolute_error(ma_train_df, ma_predict),2)
---> 12 r2 = round(r2_score(mb_train_df, mb_predict),2)
     14 plt.figure(figsize=(10, 4))
     15 plt.plot(ma_train_df.index, ma_train_df, label='Train Data')

NameError: name 'mb_predict' is not defined
```

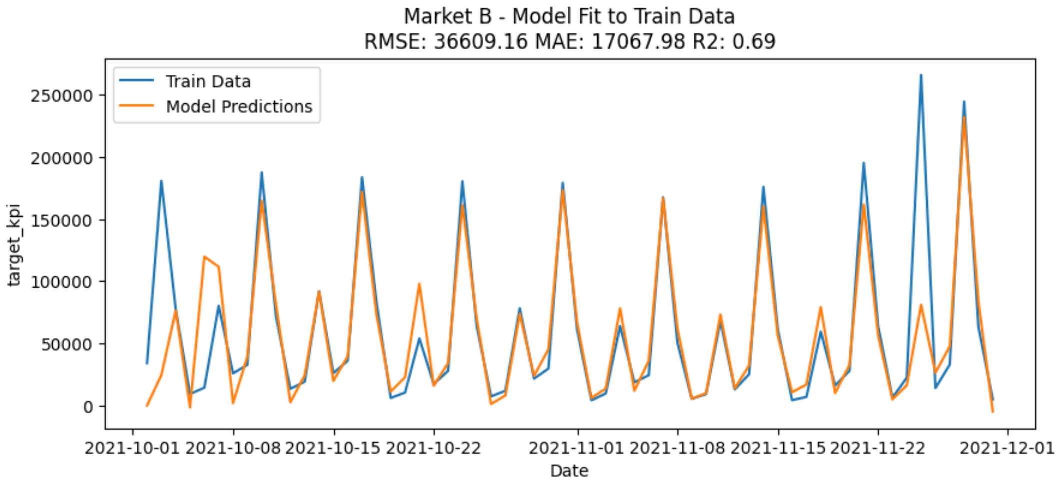
In [ ]:

```
#quick review of model diagnostics for Market A
ma_model_fit.plot_diagnostics(figsize=(15, 8))
plt.show()
```



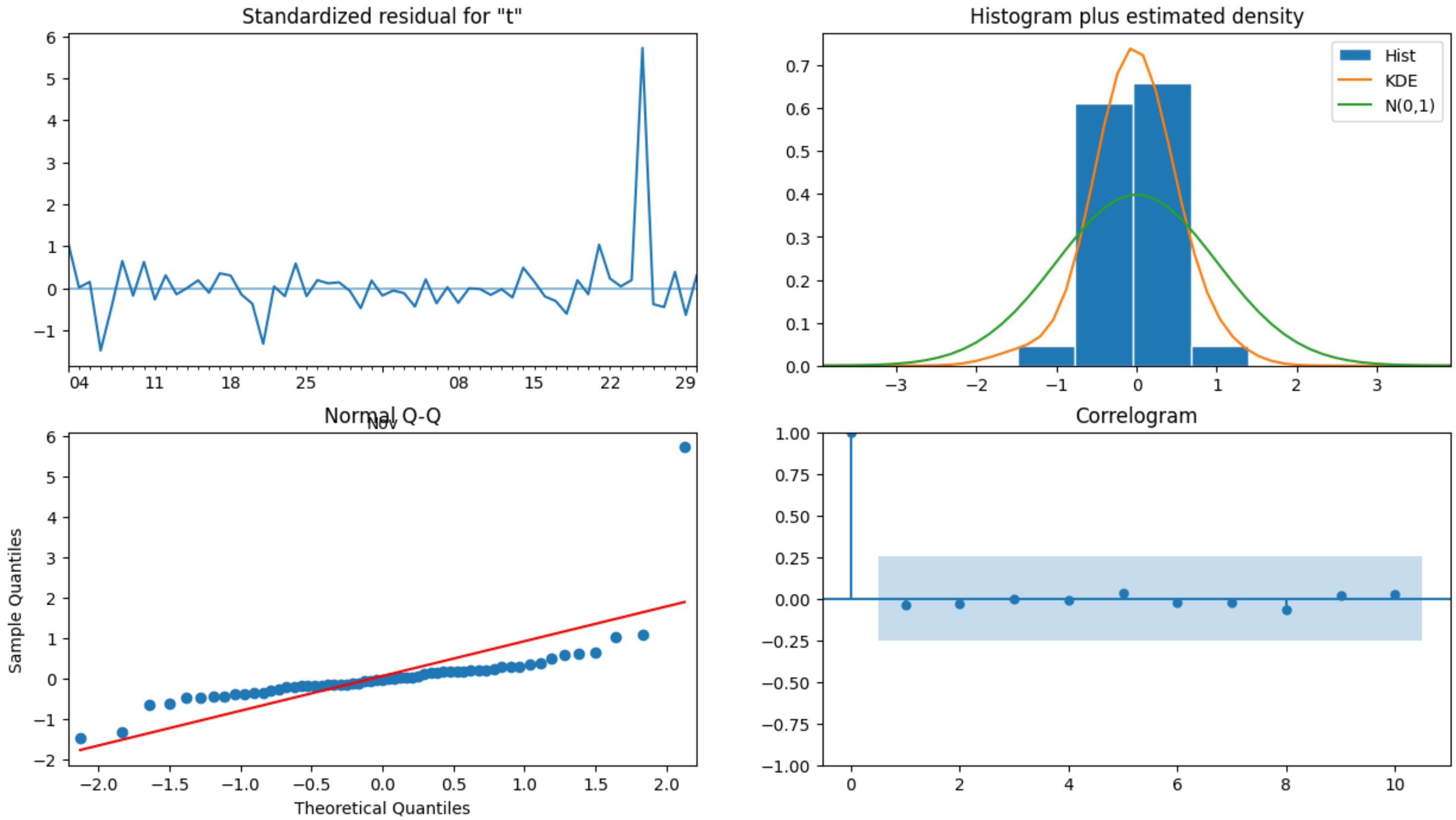
In [ ]:
```
### MARKET B MODEL ###
p, d, q = 7, 1, 7  # autoregression, differencing, moving average
mb_model = ARIMA(mb_train_df, order=(p, d, q))
mb_model_fit = mb_model.fit()

mb_predict = mb_model_fit.predict(start=mb_train_df.index[0], end=mb_train_df.index[-1])

mse = round(mean_squared_error(mb_train_df, mb_predict),2)
ma_rmse = round(np.sqrt(mse),2)
mae = round(mean_absolute_error(mb_train_df, mb_predict),2)
r2 = round(r2_score(mb_train_df, mb_predict),2)

plt.figure(figsize=(10, 4))
plt.plot(mb_train_df.index, mb_train_df, label='Train Data')
plt.plot(mb_predict.index, mb_predict, label='Model Predictions')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title("Market B - Model Fit to Train Data\n" + f'RMSE: {ma_rmse} MAE: {mae} R2: {r2}')
plt.legend()
plt.show()
```

```
/Users/patrick/Documents/Sports Analytics/prizepicks/venv/lib/python3.11/site-packages/statsmodels/tsa/statespace/sarimax.py:966: UserWarning: Non-stationary starting autoregressive paramet
ers found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
/Users/patrick/Documents/Sports Analytics/prizepicks/venv/lib/python3.11/site-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge.
Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```



In [ ]:

```
#quick review of model diagnostics for Market B
mb_model_fit.plot_diagnostics(figsize=(15, 8))
plt.show()
```



Model Fit Summary

write a quick summary of how well the models fit to the train data

Market A

After experimenting and landing on a 7-day rolling average, its performance increased dramatically in fitting the data. It explains 69% of the variance in the data, with a Mean Average Error of nearly 13,500. It was slow to adjust as the data increased drastically about a third of the way into the sample size. After that, it was closer but has room for improvement. It also missed the decrease around Thanksgiving.

Market B

The theme for missing Thanksgiving is the opposite for this one. The spike on a Thursday was not anticipated and likely led to most of the degradation in performance metrics. Its Mean Average Error was worse than Model A but visually looks more in sync. This one also explains 69% of the variance. I was pleased with where it ended up in the time I spent.

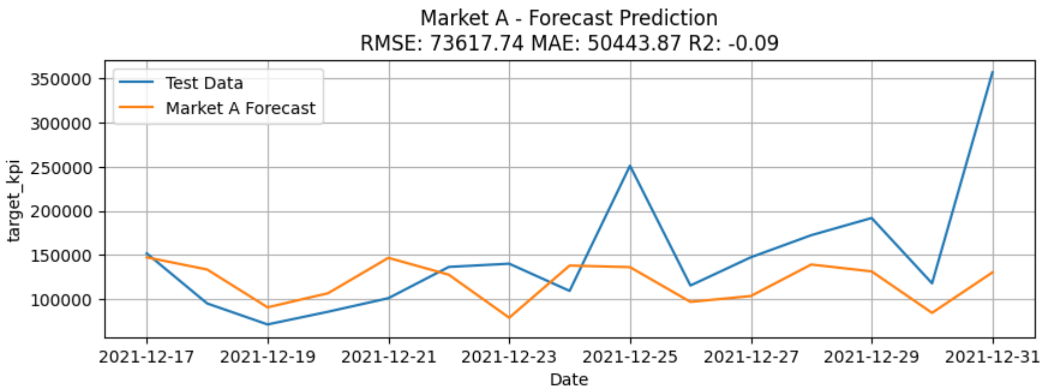
Model Forecasting

using the model - make a 14 day prediction for both Market A & Market B

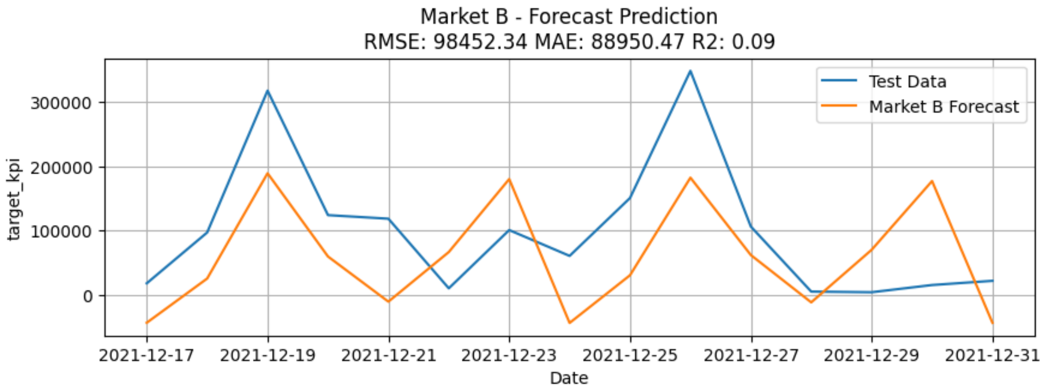
This is the area that was the sneakiest, as there are 16 missing days between the end of the training data and the beginning of the test data. I accounted for it in each of my forecasts, and there is definitely room to improve the code I wrote.

Additionally, I am making the assumption that the "14-day prediction" is meant to be the same length as the test data, which happens to be 15 days. With this in mind, as well as the missing days, the actual forecast length of days is 31. I truncated the forecast data below to match the length of the test data.

In [ ]:
```python
# MARKET A #
missing_dates = 16
ma_forecast = ma_model_fit.forecast(steps=len(ma_test_df)+missing_dates)
market_a_fore = ma_forecast[-len(ma_test_df):]

mse = round(mean_squared_error(ma_test_df, market_a_fore),2)
ma_rmse = round(np.sqrt(mse),2)
mae = round(mean_absolute_error(ma_test_df, market_a_fore),2)
r2 = round(r2_score(ma_test_df, market_a_fore),2)


plt.figure(figsize=(10, 3))
plt.plot(ma_test_df.index, ma_test_df, label='Test Data')
plt.plot(market_a_fore.index, market_a_fore, label='Market A Forecast')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title("Market A - Forecast Prediction\n" + f'RMSE: {ma_rmse} MAE: {mae} R2: {r2}')

plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```python
# MARKET B #
missing_dates = 16
mb_forecast = mb_model_fit.forecast(steps=len(mb_test_df)+missing_dates)
market_b_fore = mb_forecast[-len(mb_test_df):]

mse = round(mean_squared_error(mb_test_df, market_b_fore),2)
ma_rmse = round(np.sqrt(mse),2)
mae = round(mean_absolute_error(mb_test_df, market_b_fore),2)
r2 = round(r2_score(mb_test_df, market_b_fore),2)

plt.figure(figsize=(10, 3))
plt.plot(mb_test_df.index, mb_test_df, label='Test Data')
plt.plot(market_b_fore.index, market_b_fore, label='Market B Forecast')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title("Market B - Forecast Prediction\n" + f'RMSE: {ma_rmse} MAE: {mae} R2: {r2}')

plt.legend()
plt.grid(True)
plt.show()
```



Model Performance Analysis

compare you predictions to the forecast_test.csv dataset - write a short analysis of the models performance on the unseen data

Market A

This model visually performed okay outside of two outliers, Christmas Day and New Year's Eve Day. I'm being generous due to the lack of data that immediately preceded the test data range. Assuming this market is NBA, Christmas is notoriously a big day for the league with multiple key matchups. With the outliers, the model does not explain much variance at all, with a score of -9%. Woof. The Mean Average Error is ~55,000, which is a large increase from how the model performed with fitting the training data.

Market B

The NFL market started experiencing large growth within the two weeks of test data we have. The model did not account for this and did not perform great. The model explains roughly 9% of the variance, and the Mean Average Error is ~98,500, nearly 2.5-3x worse than the training data fit. My gut tells me that these increases might have been predicted better with the missing data (had to bring it up again!) and with perhaps bringing in more data around the growth and user acquisition of the app.

Reflection

what would you do different if you had more time? what is missing from this model?

What's missing? Data! 16 days to be specific.

Differently with more time? A list of my ideas:

- Model selection - experiment with different time series models for each of the markets separately. Adjusting for seasonality would be a good start.
- Fine turning of parameters used for the model (p, d, q)
- cross-validation/grid search for parameters (like the above)
- Explore other metrics (MAPE as an example)
- Perform an in-depth review of the plot diagnostics to help get a better understanding of the model performance
- Clean up all the code to run more efficiently and be more readable!
- Remove outliers and see how the models performs
- User/Acquisition data to anticipate growth

Guess Work

what sports do you think Market A & Market B respresent (a market is all of our offerings for a given sport)? what metric do you think the target KPI represents? why?

Market A: NBA (slow ramp up and then takes off! Christmas Day huge numbers)

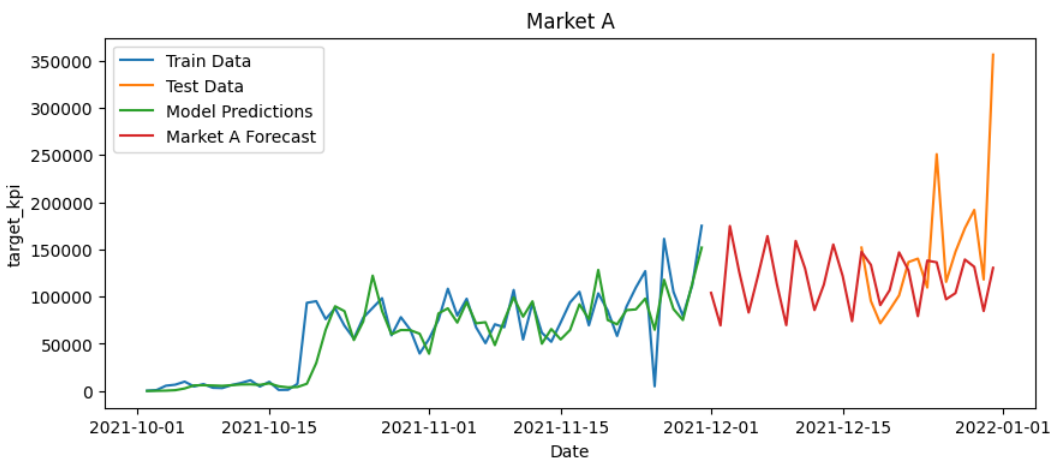
Market B: NFL (Sunday, Monday, Thursday spikes)

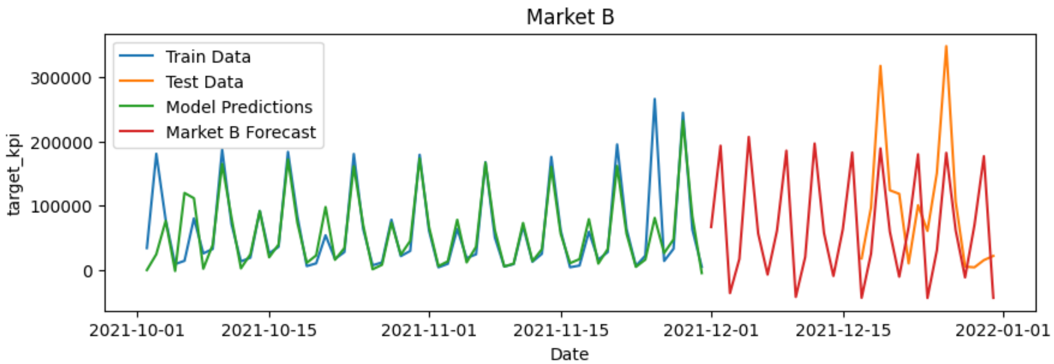
Target KPI: My first guess is the Amount of Money ($) Wagered. My thought here is that the fall of 2021 roughly puts the company at 3ish years old, and those amounts that are reached during peak winter months for NBA and NFL would (likely) be very healthy amounts of money wagered on such a young platform. My second guess would be the Number of Wagers, which would mean that the amount of money being wagered is multiples higher than my first guess.

Grand Finale

In [ ]:

```
#Everything, everywhere, all at once. heh.

### MARKET A MODEL ###

plt.figure(figsize=(10, 4))
plt.plot(ma_train_df.index, ma_train_df, label='Train Data')
plt.plot(ma_test_df.index, ma_test_df, label='Test Data')
plt.plot(ma_predict.index, ma_predict, label='Model Predictions')
plt.plot(ma_forecast.index, ma_forecast, label='Market A Forecast')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title('Market A')
plt.legend()
# plt.show()

# MARKET B #
plt.figure(figsize=(10, 3))
plt.plot(mb_train_df.index, mb_train_df, label='Train Data')
plt.plot(mb_test_df.index, mb_test_df, label='Test Data')
plt.plot(mb_predict.index, mb_predict, label='Model Predictions')
plt.plot(mb_forecast.index, mb_forecast, label='Market B Forecast')
plt.xlabel('Date')
plt.ylabel('target_kpi')
plt.title('Market B')
plt.legend()
plt.show()
```





Talk soon! - Patrick

end homework