Lesson 21 – Time Series Analysis

**Questions for Mentor:**

**Introduction to Time Series Analysis in Python continued:**

* AR and MA models
* MA(1) model is a moving average model with autocorrelation of 1
* ARMA model is a combination of an AR and MA model
* Cointegration
  + Two variables are both random walk and not forecastable, but together they are forecastable
  + Example of owner walking a dog. Each are random walk, but together are forecastable (dog too far behind or ahead, gets pulled back)
* What types of series are cointegrated?
  + Economic substitutes
    - Heating oil and natural gas
    - Platinum and palladium
    - Corn and wheat
    - Corn and sugar
    - Bitcoid and etherium
  + Competitors?
    - Coke and pepsi
    - Apple and blackberry? – dog broke the leash and ran away from the owner
  + Regress random walks and get slope c
  + Run augmented dickey-fuller on V1 -cV2 to test for random walk
  + Can use statsmodels coint function to combine steps
* ARIMA Model
  + Parameters are:
    - p – the number of lag observations included in the model
    - d – the number of times the raw observations are differenced
    - q – the size of the moving average window

**Best of the Best Models:**

* ACF and PACF
  + Autocorrelation function and partial autocorrelation function
  + Lag 1 autocorrelation is the time series correlation with itself offset by one step
    - Same goes for lags 2, 3 etc
  + ACFs small and inside shaded region, they aren’t statistically significant
  + Graphical user interface, text, application

    Description automatically generated
  + Functions in statsmodels package
    - Plot\_acf, plot\_pacf
  + Time series must be stationary before plotting acf and pacf
* AIC and BIC
  + Can’t use ACF and PACF when MA and AR are both non-zero
  + Akaike information criteria
    - Lower AIC indicates better model
    - AIC likes lower order better – less overfitting
  + Bayesian information criterion
    - Very similar to AIC
    - Lower BIC is better model
    - BIC likes to choose simpler model with lower order
    - Penalizes overcomplex models
  + Difference is how much they penalize model complexity
    - BIC favors simpler more than AIC
  + AIC is better at choosing predictive models
  + BIC is better at choosing explanatory models
* Model diagnostics
  + Residuals
    - .resid attribute
  + .plot\_diagnostics() will show 4 plots to show diagnostics
* Box-Jenkins method
  + Identification
    - Stationary?
      * .plot()
      * Adfuller()
    - What differencing will make it stationary?
    - Which transforms will make it stationary?
    - What p and q will be best?
  + Estimation
    - Estimate AR and MA coefficients
    - Can use AIC and BIC to narrow down candidates
  + Model diagnostics
    - .plot\_diagnostics()
    - .summary()
  + Decide – model okay?
    - If residuals are not as they should be, we need to rework the model
    - If it is okay – go to production

**Machine Learning with Time Series:**

* Time series
  + array of data that contains the time series itselt
  + points that correspond with time point
* period of time series = time between time points
* ML allows us to predict the future
* Main steps
  + Feature extraction
  + Model fitting
  + Prediction and validation
* Machine Learning and time series data
* Classifying a time series
* Audio files
  + Take sound data, assign timeseries
* Can smoth time series using rolling window
* .rolling() takes rolling mean with specified window size
* Tempogram using librosa library for audio analysis
* Spectrogram
  + Fourier Transforms (FFT)
  + Short term fourier transforms (STFT)
  + To calculate spectrogram, square the STFT
  + Spectral centroid and bandwith
    - Show pattern in spectrogram
* Predicting data over time
  + Real world data is often messy
  + Often due to failing sensors, human error, DB failures
  + Can use interpolation to interpolate between start and stop points of empty data
  + Rolling window can smooth time series change
  + Remove/replace outliers
    - Center around zero
    - Establish thresholds to remove abberations and replace with median value (or something like it)
* Creating features over time
  + .aggregate can calculate many methods of a feature at once
    - Can use partial function, allows you to use part of function
  + Np.percentile gives you specified percentile of the array
  + Date based features
    - More human features like day of the week etc
* K-fold cross validation
* One cross validation technique that is specifically for time series data
  + TimeSeriesSplit()
* Stationary signal
  + Don’t change statistical properties over time
    - Mean, std, trends etc
  + Most time series are non-stationary to some extent
  + CV to quantify stability
  + Bootstrapping
    - def bootstrap\_interval(data, percentiles=(2.5, 97.5), n\_boots=100):
    - """Bootstrap a confidence interval for the mean of columns of a 2-D dataset."""
    - # Create our empty array to fill the results
    - bootstrap\_means = np.zeros([n\_boots, data.shape[-1]])
    - for ii in range(bootstrap\_means):
    - # Generate random indices for our data \*with\* replacement, then take the sample mean
    - random\_sample = \_\_\_\_
    - bootstrap\_means[ii] = random\_sample.mean(axis=0)
    - # Compute the percentiles of choice for the bootstrapped means
    - percentiles = \_\_\_\_(bootstrap\_means, percentiles, axis=0)
    - return percentiles
* Advanced time series feature extraction – tsfresh library