Specialized Models:

Time Series and Survival Analysis

Course Project

Evgeny Zorin

5. December 2021

Air Passengers Prediction

Abstract

Predicting the number of passengers traveling by air is an important topic in the aviation business and travel economics because it allows airlines to better understand what strategy they should develop to meet passenger demand for air travel. Accurate prediction is the key to success for airlines in developing and improving their business models.

Main Objective

The main objective of the project is to build a model that will be able to predict the demand for air travel (passenger traffic) in the United States for the next 12 months based on data from the past 12 years.

Project Workflow

1. Data Loading and Exploration
    - Data Summary
    - Data Visualization
2. Data Pre-processing
    - Stationarity Analysis
    - Differencing
    - Train Test Split
3. Modeling
    - Holt-Winters Model
    - SARIMA Model
    - FBProphet Model
    - Vanilla LSTM Model
    - Model Evaluation
    - Future Forecast
4. Key Findings and Insights
5. Next Steps

Data Loading and Exploration

Data Summary

The data was downloaded from Kaggle and it contains 144 monthly totals of a US air passengers from 1949 to 1960.

The data consists of 2 features:

- **Month** - Month of the year
- **Passengers** - Total number of passengers travelled on the given month

The Month feature is represented as a string, so it will be converted to *datetime* format and set as an index. A sample of prepared data is shown in *Table 1*.

Table 1: Data Sample

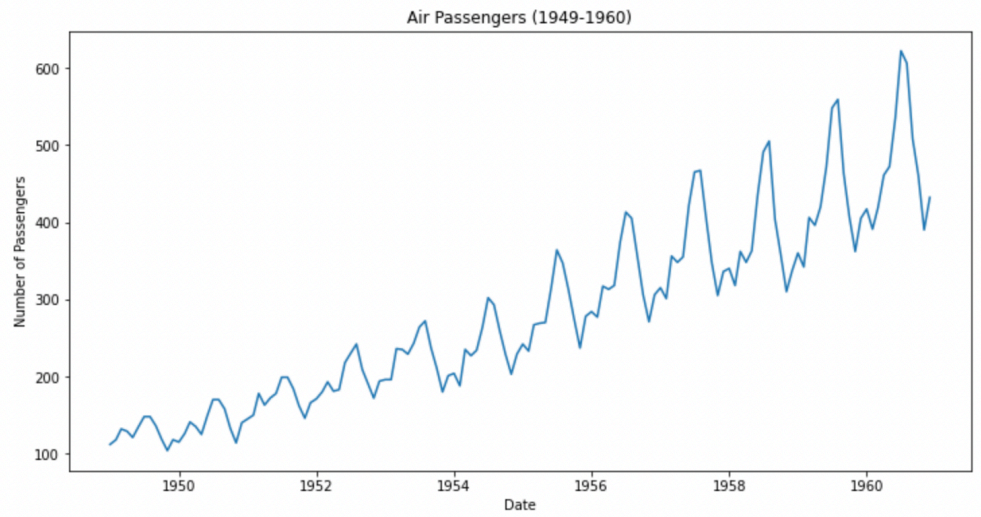
Month	Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Data Visualization

Data visualization helps to better understand what kind of data will be used in the project, as well as identify the steps that should be taken to prepare the data for the modeling and forecasting process.

The plot (*Figure 1*) shows that the data has a strong upward trend, meaning the number of passengers increases from year to year, and also a seasonality that is explained by increasing the number of passengers in the summer periods and decreasing in the winter periods.

Figure 1: Data Visualization



Data Pre-processing

Stationarity Analysis

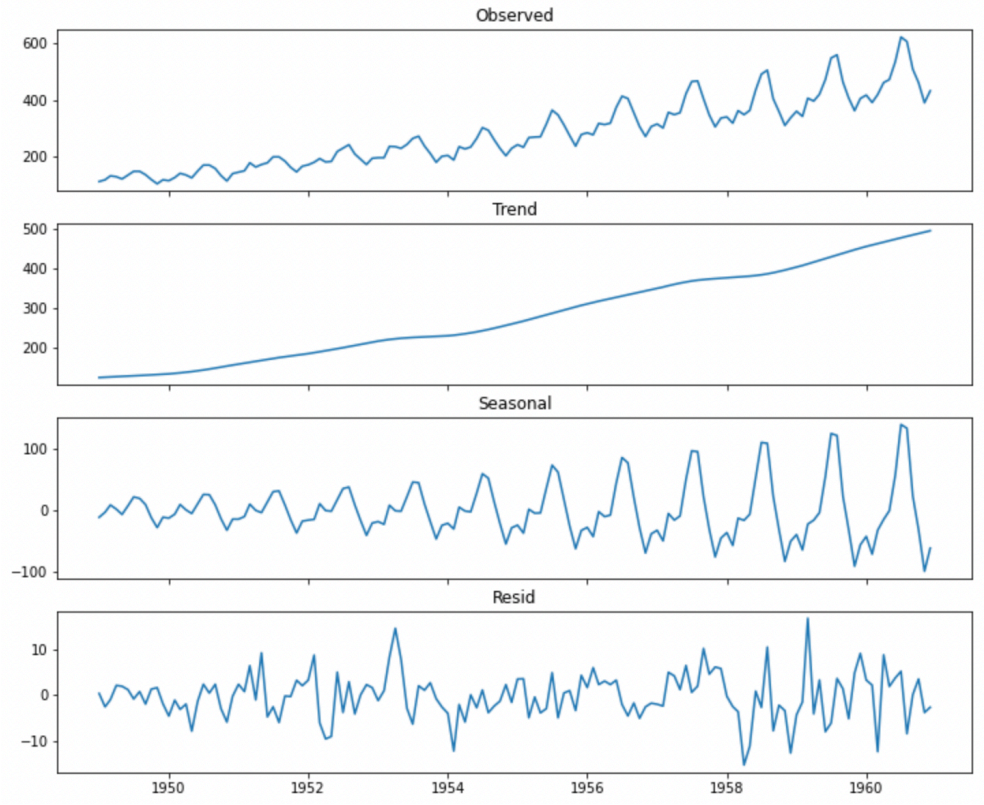
Stationarity is a key part of time series analysis. Stationarity means that the way the time series data changes is constant. A stationary time series will not have any trends or seasonal patterns. Stationarity should be checked because it not only simplifies time series modeling, but it is also a basic assumption in many time series analysis methods. In particular, stationarity is assumed for a wide range of time series forecasting methods, including autoregressive moving average (ARMA), ARIMA, and seasonal ARIMA (SARIMA).

The Augmented Dickey-Fuller (ADF) test will be used to check the stationarity of the data. This test generates a p-value that will allow rejecting or not rejecting the null hypothesis that the data are non-stationary. If the null hypothesis is rejected, the alternative hypothesis stating that the data are stationary will be accepted.

Trend decomposition is another useful way to visualize trends in time series data.

The decomposed data plot (*Figure 2*) and the ADF test (p-value = 0.9918) confirm the non-stationarity of the data.

Figure 2: Data Decomposition



Differencing

One of the common approaches to make data stationary is to compute the differences between consecutive observations. This is known as differencing.

Differencing can help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

After differencing the data, the ADF test shows that the p-value is less than 0.05 (p-value = 0.0466), so it can be claimed with 95% confidence that the differenced data is stationary.

Train Test Split

As can be seen on the decomposition plot, the seasonality is represented by 12 months, so this period will also be used as the prediction window size, as well as the window size for the SARIMA and LSTM models. In other words, the training data will be 132 months, and the test data will be 12 months.

Modeling

Holt-Winters Model

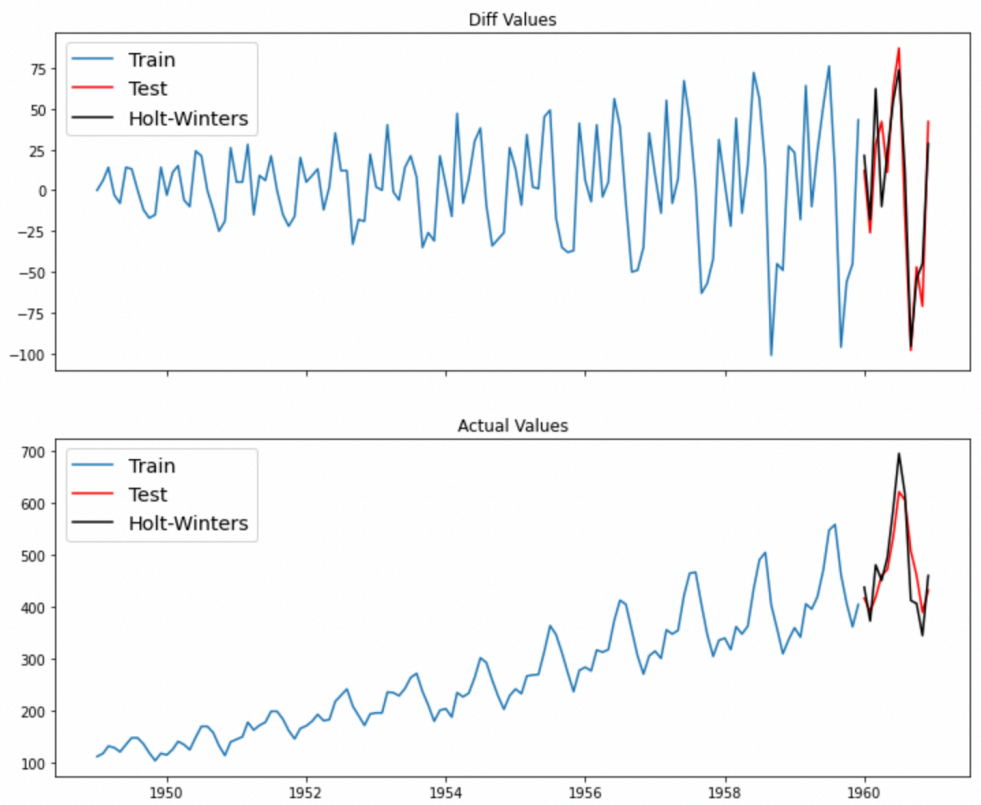
The Holt-Winters method uses exponential smoothing to encode lots of values from the past and use them to predict "typical" values for the present and future. Exponential smoothing refers to the use of an exponentially weighted moving average (EWMA) to "smooth" a time series.

The three aspects of the time series behavior — value, trend, and seasonality — are expressed as three types of exponential smoothing, so Holt-Winters is called triple exponential smoothing. The model predicts a current or future value by computing the combined effects of these three influences.

The prediction plots (*Figure 3*) show that in general the model performs quite well. The differenced prediction values on the test period exactly repeat the last training 12-month period. The actual values show that the model has a high error at the peaks of minimum and maximum values.

RMSE indicates that the accuracy of the model is not high enough to accurately predict the future, but it is good enough to get a general picture of what will happen in the future.

Figure 3: Holt-Winters Model Prediction



SARIMA Model

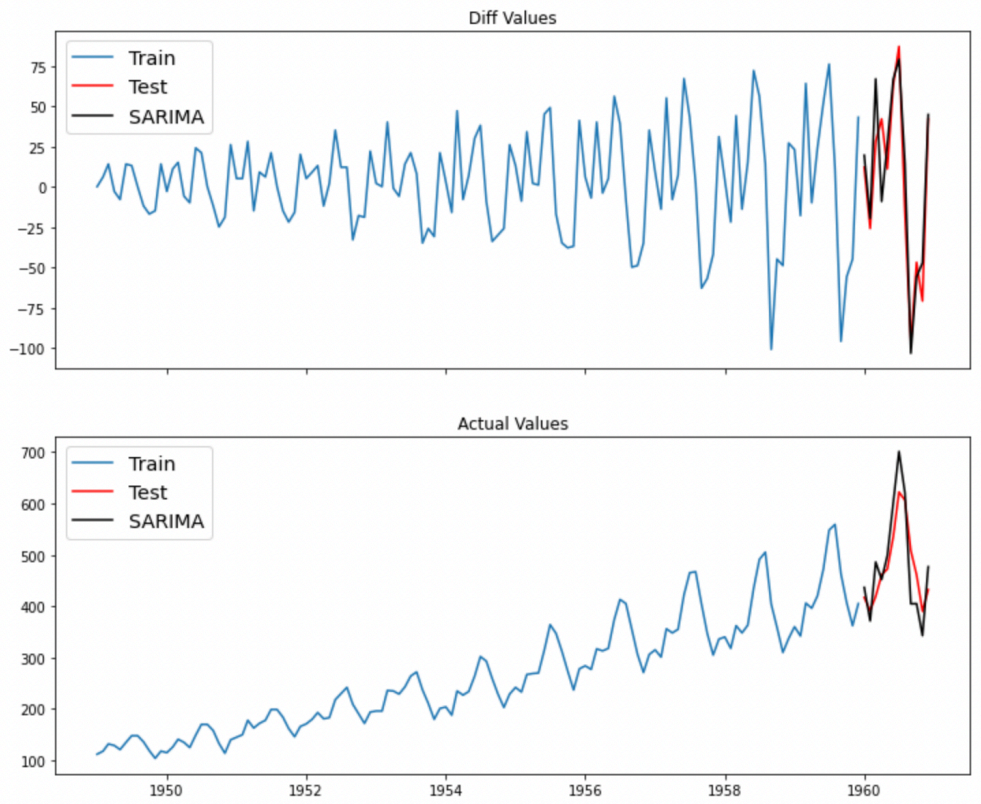
Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of the ARIMA model that explicitly supports univariate time series data with a seasonal component.

It adds three new hyper-parameters to specify the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality.

The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshifts of the seasonal period.

The prediction plots (*Figure 4*), as well as the RMSE, are almost identical to the results of the Holt-Winters model, but the SARIMA model performed slightly better.

Figure 4: SARIMA Model Prediction



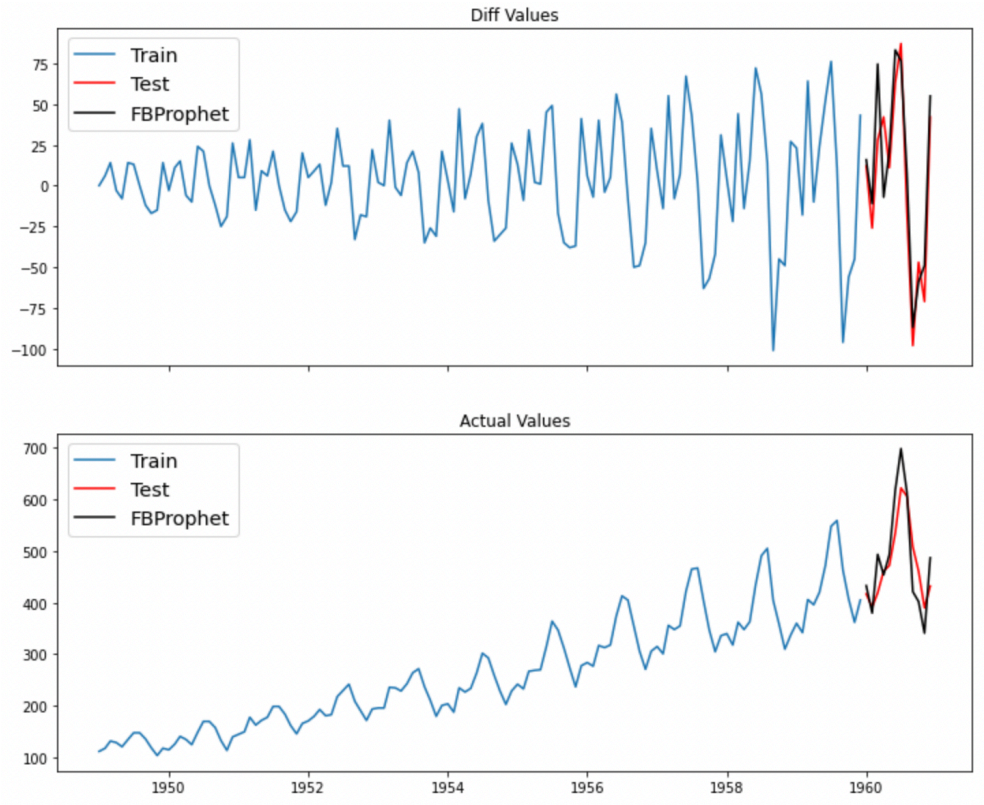
FBProphet Model

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

The input to Prophet is always a dataframe with two columns: **ds** and **y**. The **ds** (datestamp) column should be of a format *YYYY-MM-DD* for a date or *YYYY-MM-DD HH:MM:SS* for a timestamp. The **y** column must be numeric, and represents the measurement to be forecasted.

The prediction plots (*Figure 5*), as well as the RMSE, show that the FBProphet model performed worse than the previous two models.

Figure 5: FBProphet Model Prediction



Vanilla LSTM Model

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs and other models.

This project will use the Vanilla LSTM model, which is an LSTM model that has one hidden layer of LSTM blocks, and an output layer used for prediction. A Dropout layer will also be added to the model to avoid overfitting. The model summary is shown in *Figure 6*.

Figure 6: Model Summary

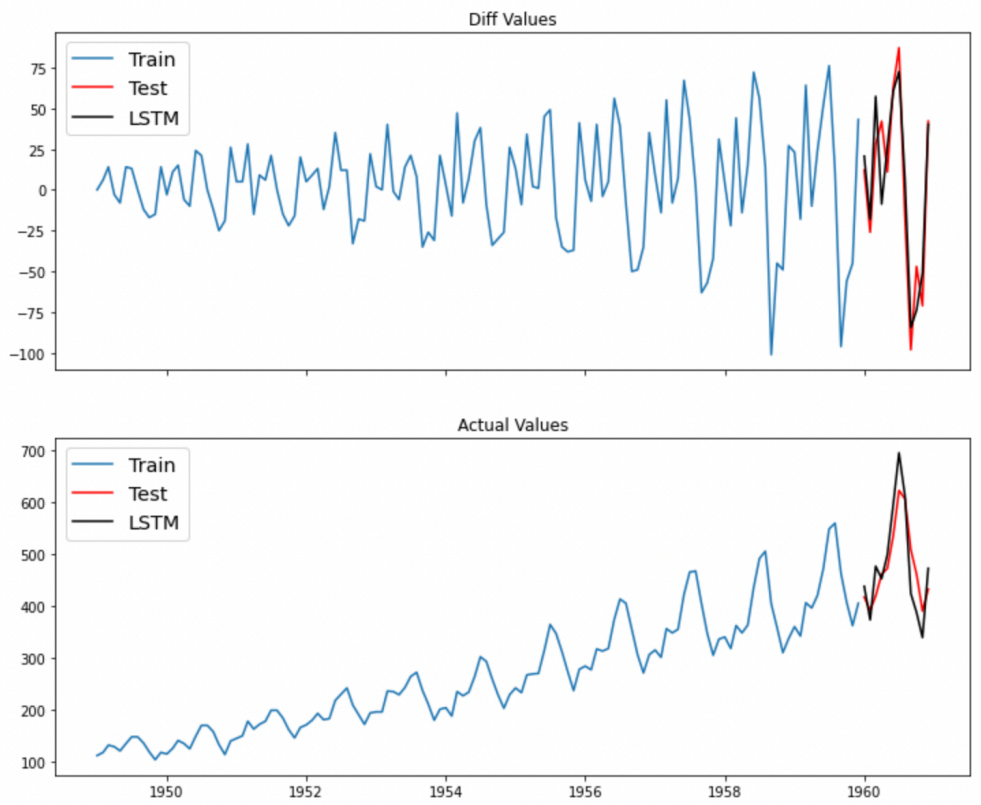
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 200)               161600

 dropout (Dropout)           (None, 200)               0

 dense (Dense)               (None, 1)                 201

=================================================================
Total params: 161,801
Trainable params: 161,801
Non-trainable params: 0
```

The data will be scaled using *MinMaxScaler* to translate each observation in the training set to the given range between zero and one.

The *TimeseriesGenerator* will be used to prepare the input data for the LSTM model, which will generate sequences of training data equal to 12 months each.

The prediction plots (*Figure 7*), as well as the RMSE, are almost identical to the results of the Holt-Winters and SARIMA models, but the Vanilla LSTM showed the best results among all the models.

Figure 7: Vanilla LSTM Model Prediction



Model Evaluation

The prediction error results on the test set for all 4 models are almost identical (*Table 2*). The FBProphet model performed the worst, while the Vanilla LSTM model performed the best.

Table 2: Prediction Error Results

	Holt-Winters	SARIMA	Prophet	LSTM
RMSE	22.517141	22.657681	23.977944	22.049379

In general, the predictions of all 4 models are not accurate enough to build a business model on them. But they are good enough to understand airline market trends.

Table 3 shows the differenced prediction values of all models relative to the test set, and *Table 4* shows the actual prediction values of all models relative to the test set. *Figure 8* visualizes the prediction values in these tables.

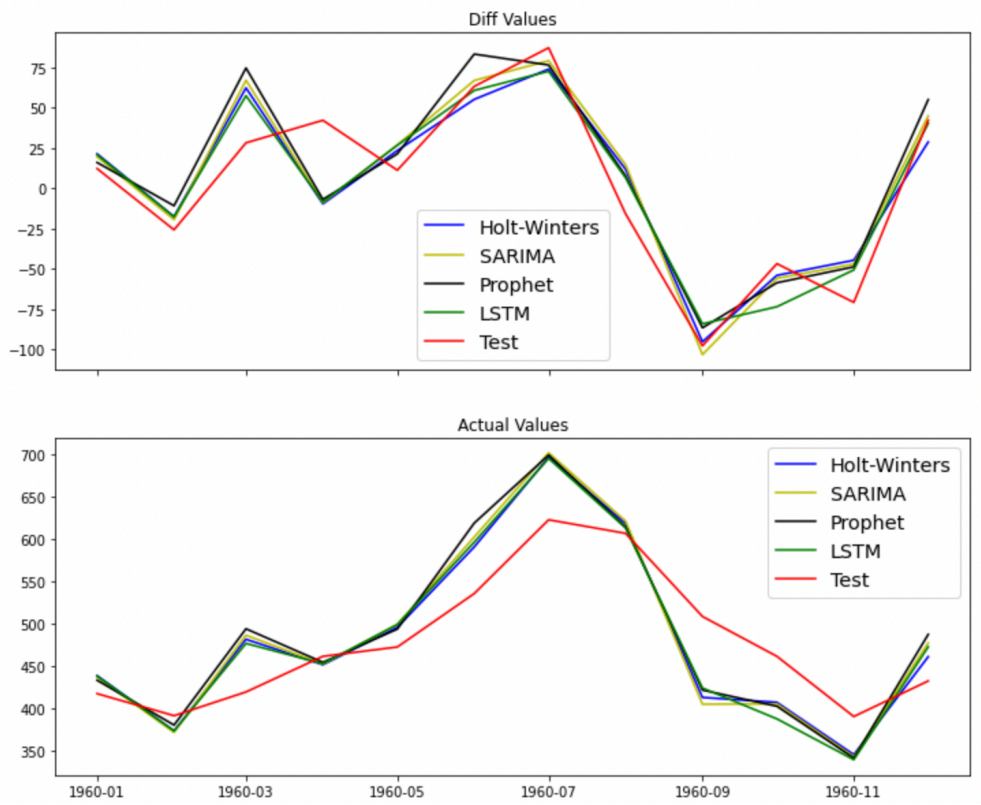
Table 3: Differenced Prediction Values

Month	Passengers	Holt-Winters	SARIMA	Prophet	LSTM
1960-01-01	12	21	19	15	20
1960-02-01	-26	-17	-19	-11	-17
1960-03-01	28	62	66	74	57
1960-04-01	42	-9	-9	-7	-8
1960-05-01	11	23	26	21	26
1960-06-01	63	54	66	83	60
1960-07-01	87	73	78	76	72
1960-08-01	-16	11	14	7	6
1960-09-01	-98	-95	-103	-86	-84
1960-10-01	-47	-54	-56	-58	-73
1960-11-01	-71	-44	-47	-48	-50
1960-12-01	42	28	44	54	40

Table 4: Actual Prediction Values

Month	Passengers	Holt-Winters	SARIMA	Prophet	LSTM
1960-01-01	417	438	436	432	437
1960-02-01	391	373	371	379	373
1960-03-01	419	481	485	493	476
1960-04-01	461	451	451	453	452
1960-05-01	472	495	498	493	498
1960-06-01	535	589	601	618	595
1960-07-01	622	695	700	698	694
1960-08-01	606	617	620	613	612
1960-09-01	508	412	404	421	423
1960-10-01	461	406	404	402	387
1960-11-01	390	345	342	341	339
1960-12-01	432	460	476	486	472

Figure 8: Model Prediction Results



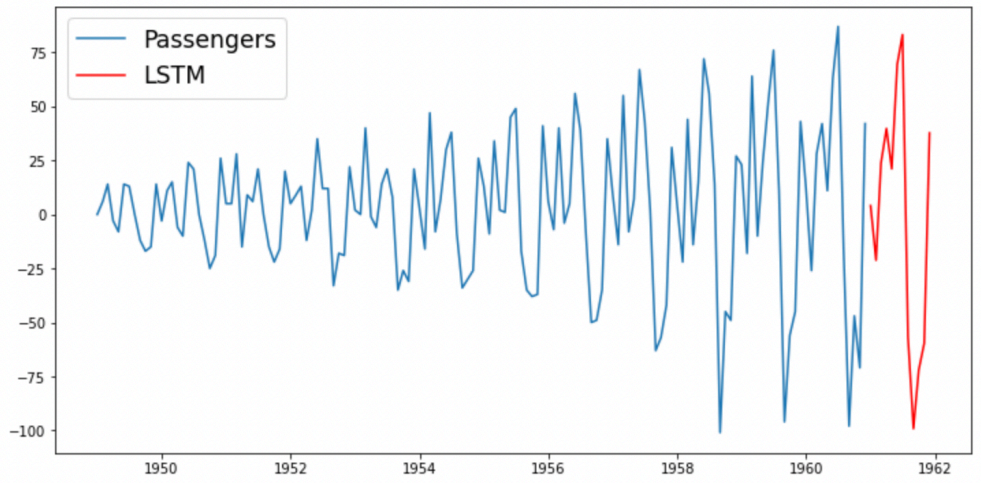
Future Forecast

The model with the lowest error, which is the Vanilla LSTM model, will be used as the model for future forecasting.

Future forecast will be performed for the next 12 months (year 1961). But this time all differenced data will be used as the train set.

Future forecast result is shown in *Figure 9*.

Figure 9: Future Forecast Result



Key Findings and Insights

- An important step in building models based on time series data is to check the data for stationarity and use only stationary data as input. When a time series is stationary, it can be easier to model. Statistical modeling methods assume or require the time series to be stationary to be effective.
- In general all the models perform quite well, but their RMSE indicate that the accuracy of the models is not high enough to accurately predict the future, but it is good enough to get a general picture of what will happen in the future.
- The Vanilla LSTM model provides the best results, but it also needs additional tuning.

Next Steps

- Probably inaccurate predictions are due to the small number of observations in the dataset, which are insufficient for efficient model training. This is especially relevant for the LSTM model, which is able to provide good results on large datasets. The next step would be to obtain more observations and expand the given dataset.
- In addition, adding more hidden layers to the Vanilla LSTM model, as well as using validation set with EarlyStopping callback to avoid overfitting, would help to get better results.
- After expanding the given data with more observations, it would be useful to use other models to train the data, such as RNN, CNN, GRU and XGBoost, before tuning the Vanilla LSTM model.

Appendix

GitHub URL:
https://github.com/evgenyzorin/IBM-Machine-Learning/tree/main/Time-Series