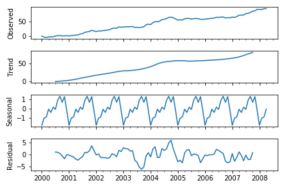
Congratulations! You passed!

Grade received 100% | Latest Submission Grade 100% | To pass 80% or higher

Go to next item

1. Realizamos un análisis del número de personas que se alojan en una cadena hotelera a lo largo de los años. Descomponemos la serie obteniendo el siguiente gráfico: 1 / 1 point



¿Qué podemos decir de los datos? Selecciona las respuestas correctas

- [ ] Hay una tendencia decreciente a lo largo de los años
- [x] Hay una marcada estacionalidad

Correct
Correcto. Esto se aprecia en los patrones repetitivos de la gráfica seasonal

- [x] Hay bastante ruido en los datos

Correct
Correcto. Se pueden observar múltiples picos diferentes a 0 en la gráfica de residual

- [ ] Por la tendencia se puede concluir que hay una estacionaridad en los datos

2. Queremos entrenar un modelo básico para predecir el número de ventas anuales de una cadena de supermercados. ¿Qué códigos y funciones debemos utilizar para generar el modelo y evaluar su desempeño? 1 / 1 point

Selecciona las respuestas correctas

- [ ] s = setup(data = train, test_data = test, target = ventas, session_id = 123)
- [x] best = compare_models( )

Correct
Correcto. Esta función nos permitirá entrenar múltiples modelos y seleccionar el mejor en base al MAE

- [x] predict_model(best)

Correct
Correcto. Esta función permite realizar una predicción con el mejor modelo y después obtener las métricas de rendimiento como MAE, MSE, RMSE, etc.

- [x] tune_model(best)

Correct
Correcto. Esta función permite optimizar el modelo mediante el ajuste de hiperparametros.

3. ¿Qué funciones realiza el siguiente código de Pycaret? 1 / 1 point

Selecciona las respuestas correctas

```
huber = create_model('huber', verbose = False)
rf = create_model('rf', verbose = False)
lar = create_model('lar', verbose = False)
lr = create_model('lr',verbose=False)
```

```
tuned_rf = tune_model(rf)
tuned_huber = tune_model(huber)
tuned_lar = tune_model(lar)
tuned_lr = tune_model(lr)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	20.5658	819.0068	28.6183	0.1724	0.1157	0.0949
1	11.0114	210.1223	14.4958	0.9153	0.0445	0.0358
2	35.6145	1742.3307	41.7412	0.5986	0.1033	0.0930
Mean	22.3972	923.8199	28.2850	0.5621	0.0878	0.0746
SD	10.1273	629.8968	11.1255	0.3044	0.0311	0.0274

```
blend_specific = blend_models(estimator_list = [tuned_rf,tuned_lar,tuned_lr,tuned_huber])
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	12.6354	230.7430	15.1902	0.7668	0.0792	0.0617
1	27.5541	1191.7621	34.5219	0.5194	0.1068	0.0844
2	21.4340	715.0835	26.7410	0.8353	0.0626	0.0515
Mean	20.5412	712.5295	25.4844	0.7072	0.0829	0.0659
SD	6.1232	392.3385	7.9420	0.1357	0.0185	0.0138

- [ ] Entrena tres modelos de Gradient Boosting regresor, CatBoost Regressor y K Neighbors Regressor
- [x] Optimiza los modelos entrenados previamente

Correct
Correcto. Para ello utiliza la función de tune_model ( )

- [ ] Los modelos entrenados se ensamblan mediante la técnica de boosting
- [x] El modelo final tiene un rendimiento aceptable, con un R2 de 0.7

Correct
Correcto. El modelo restante tiene un rendimiento aceptable, lo que se observa en la última tabla de métricas donde el R2 es de 0.7, y los valores para el resto de las métricas son bastante reducidos.

4. Hemos entrenado un modelo para predecir el número de pasajeros a lo largo de los años. Con este modelo queremos realizar predicciones para valores futuros que van desde el 1961-01-01 hasta el 1965-01-01. 1 / 1 point

¿Qué códigos y funciones debemos incluir para conseguir esto?

Selecciona las respuestas correctas

- [ ] Debemos utilizar la función predict_model(best, data=data) donde best es el modelo y data es el dataset completo de train más test
- [x] Debemos crear un nuevo dataframe con

  future_dates = pd.date_range(start = '1961-01-01', end = '1965-01-01', freq = 'MS')

future_df = pd.DataFrame()

future_df['Month'] = [i.month for i in future_dates]

Correct
Correcto. De esta manera en dates almacenamos los meses futuros sobre los que queremos predecir y después añadimos esas fechas al dataframe future_df

- [x] Debemos añadir las mismas variables que las utilizadas en el entrenamiento al dataframe de fechas futuras

Correct
Correcto. Si para entrenar el modelo hemos utilizado las variables, mes, año y series para la predicción debemos generarlas también. Un ejemplo seria utilizar el código:

future_df['Month'] = [i.month for i in future_dates]

future_df['Year'] = [i.year for i in future_dates]

future_df['Series'] = np.arange(145,(145+len(future_dates)))

- [ ] Debemos utilizar la función compare_models(sort = 'MAE')

5. ¿Qué técnica de combinación de modelos podemos utilizar para combinar múltiples modelos y después promediar las predicciones individuales de todos ellos para formar una predicción final? **1 / 1 point**

Selecciona el código que realiza dicha función

- ( ) Los modelos no se pueden combinar
- (•) blend_models(estimator_list = [model1, model2, model3, model4])
- ( ) tune_model(model)
- ( ) ensemble_model(best)

Correct
Correcto. La técnica de blending permite crear múltiples modelos y luego promediar las predicciones individuales para formar una predicción final. Esta técnica se aplica con la función blend_models( ) de Pycaret

6. ¿Qué funciones o fases de preprocesamiento son indispensables a la hora de entrenar modelos para predecir series temporales con Pycaret? **1 / 1 point**

Selecciona las respuestas correctas

- [ ] Añadir el parámetro transform_target = True a la función de setup( )
- [x] Añadir el parámetro fold_strategy = 'timeseries' a la función de setup( )

Correct
Correcto. Mediante este parámetro indicamos a Pycaret que los modelos de regresión que entrene estén enfocados a la predicción de series temporales y con ello se mantiene la configuración necesaria para series temporales, como es que no se altere el orden de los datos

- [x] Aplicar el siguiente código pd.to_datetime(data['Date'])

Correct
Correcto. Mediante la transformación explicita de la variable de Date a formato de fecha nos aseguramos de que el formato sea el correcto y que se pueden realizar predicciones

- [ ] Añadir el parámetro data_split_shuffle = True a la función de setup( )

7. Tenemos el siguiente fragmento de código de Pycaret. **1 / 1 point**

```
from pycaret.regression import *

all_ts = data['time_series'].unique()

all_results = []
final_model = {}

for i in tqdm(all_ts):

    df_subset = data[data['time_series'] == i]

    # initialize setup from pycaret.regression
    s = setup(df_subset, target = 'sales', train_size = 0.95,
              data_split_shuffle = False, fold_strategy = 'timeseries', fold = 3,
              ignore_features = ['date', 'time_series'],
              numeric_features = ['day_of_year', 'year'],
              categorical_features = ['month', 'day_of_week'],
              silent = True, verbose = False, session_id = 123)

    # compare all models and select best one based on MAE
    best_model = compare_models(sort = 'MAE', verbose=False)

    # capture the compare result grid and store best model in list
    p = pull().iloc[0:1]
    p['time_series'] = str(i)
    all_results.append(p)

    # finalize model i.e. fit on entire data including test set
    f = finalize_model(best_model)

    # attach final model to a dictionary
    final_model[i] = f

    # save transformation pipeline and model as pickle file
    save_model(f, model_name='trained_models/' + str(i), verbose=False)
```

¿Qué funciones o tareas realiza?

Selecciona las respuestas correctas

- [ ] Optimiza el modelo ajustando los hiperparametros
- [x] Itera a lo largo de las diferentes series temporales "time_series" entrenando múltiples modelos y seleccionando el mejor modelo para esa serie

Correct
Correcto. El código itera a lo largo de las diferentes series temporales "time_series" entrenando múltiples modelos con los datos de esa serie y seleccionando el mejor modelo para esa serie en base al MAE

- [ ] El modelo resultante está entrenado con los datos de train
- [x] El modelo se guarda en formato de pickle

Correct
Correcto. Esto se realiza mediante la función de save_model(f, model_name='trained_models/' + str(i), verbose=False)