

Actividad Guiada 2 (Extr): DengAI. Predicting Disease Spread - Predicción

Nombre: Daniel Portugal Revilla

Objetivos: Predicción sobre los casos de dengue en una semana a partir de los datos meteorológicos de la misma. La competición mide el error cometido en la predicción.

Técnica a utilizar: Técnicas de predicción de las vistas en la asignatura.

Tecnología: Libre, debe justificarse en la memoria a entregar.

Introducción y definición del problema

La competición le reta a predecir el número de casos de dengue que se notifican cada semana en San Juan (Puerto Rico) e Iquitos (Perú) utilizando los datos ambientales recogidos por diversos organismos del Gobierno Federal de los Estados Unidos. Los datos proporcionados tienen características como la temperatura, la humedad y la precipitación máximas semanales. El aumento de las precipitaciones también debería contribuir al aumento de los mosquitos y, por lo tanto casos de fiebre del dengue. Con una gran cantidad de datos climáticos y otros factores, vamos a calcular los grandes contribuyentes y predecir los resultados de los datos proporcionados más adelante.

Descripción de los datos

1. 3 conjuntos de datos proporcionados por Driven Data
2. 22 características:
 - a. City, year, weekofyear,....
 - b. Los datos climáticos son los más comunes, es posible eliminar las características correlacionadas

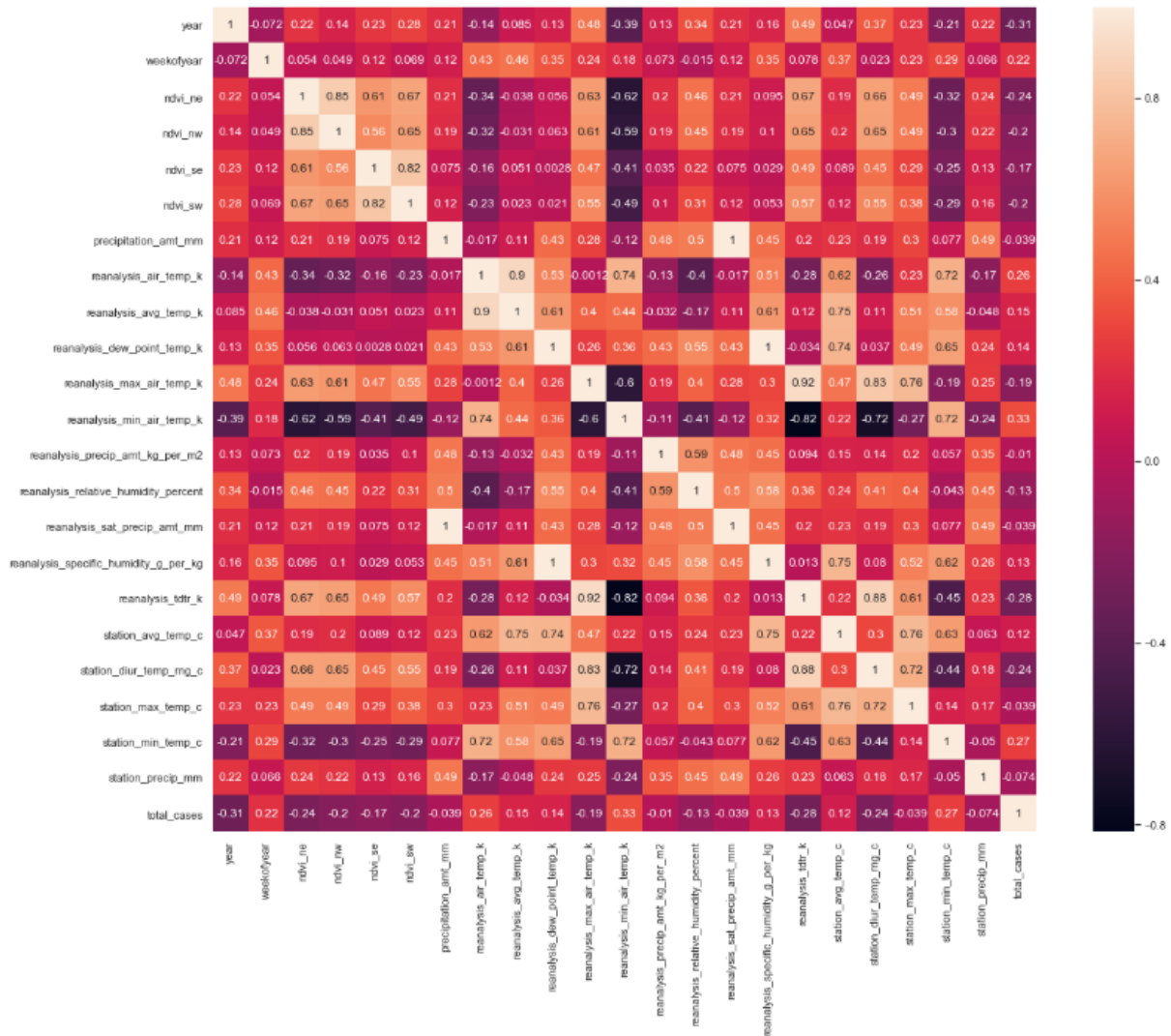
```
train = pd.read_csv('./Data/dengue_features_train.csv', encoding='utf-8')
test = pd.read_csv('./Data/dengue_features_test.csv', encoding='utf-8')
labels = pd.read_csv('./Data/dengue_labels_train.csv', encoding='utf-8')
```

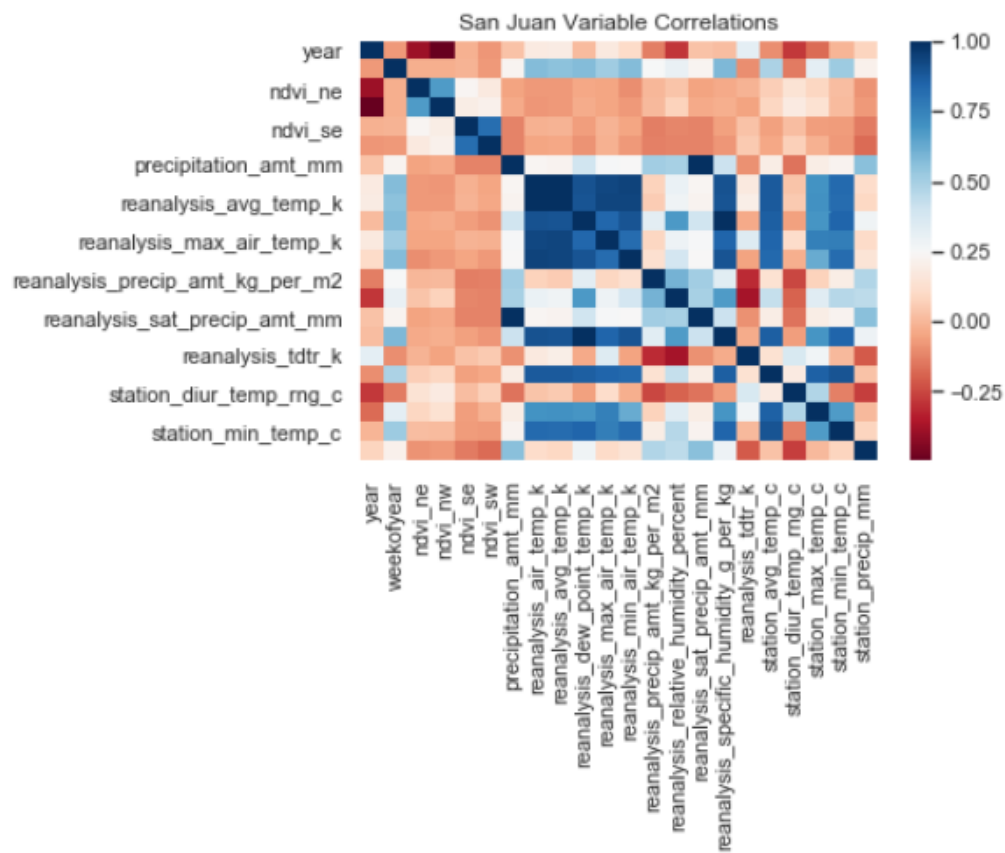
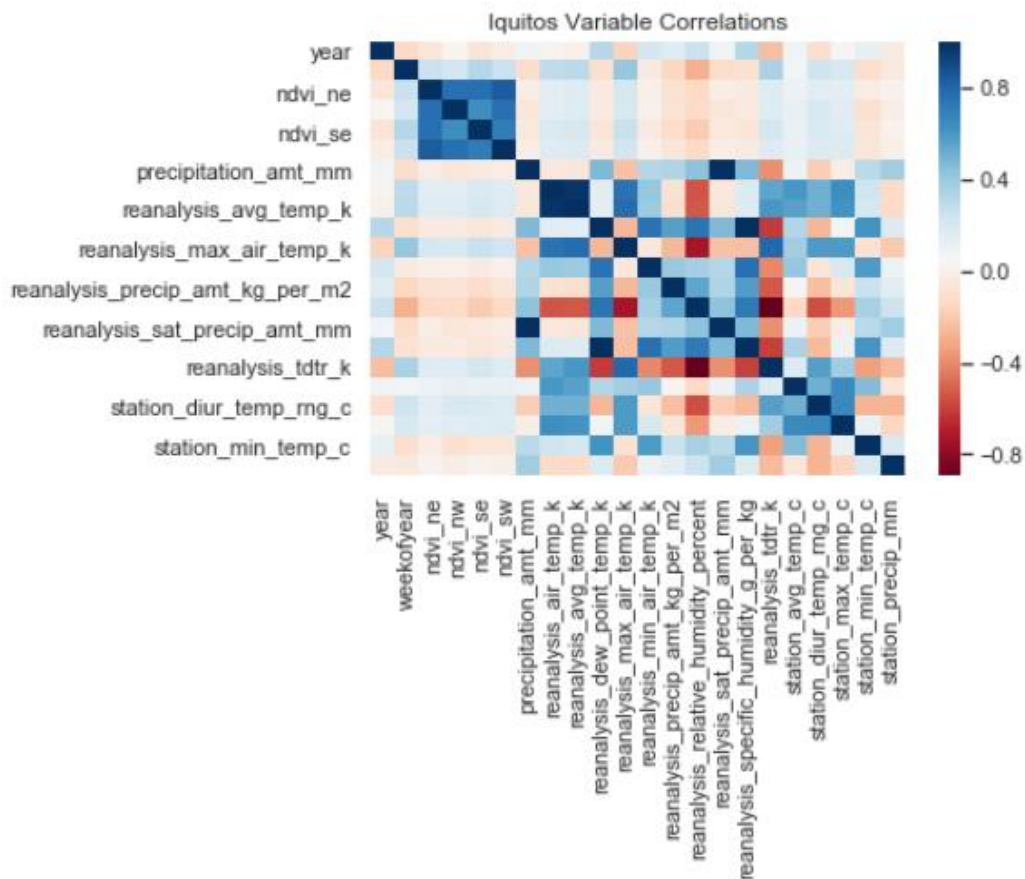
Correlación de variables

visualizamos la correlación entre variables y con la variable destino "total_cases"

```
sns.set()  
fig, ax = plt.subplots(figsize=(20,15))  
sns.heatmap(dfJoined.corr(), square=True, annot=True, ax=ax)
```

<matplotlib.axes._subplots.AxesSubplot at 0x22f1f934978>





Los dos mapas de calor de abajo describen la correlación interna del conjunto de datos para ambas ciudades. Mayor Las correlaciones de los datos implican principalmente datos climáticos. Para los datos de San Juan, la prominente Los ejemplos son los registros de datos relacionados con la temperatura. Esto tiene sentido ya que el clima de la ciudad varían muy poco. Donde los datos tienden a no correlacionarse es con la ubicación y la humedad. Los mismos resultados pueden decirse para Iquitos. Sin embargo, los datos de localización de Iquitos están mucho más correlacionados, pero la correlación para los datos climáticos es más concentrada y dispersa. La implicación para ambos es que algunos de los datos climáticos pueden ser eliminados ya que hay una baja variabilidad en ellos.

Random Forests

Nos decidimos por esta técnica para aprender más sobre los árboles de decisión. Sklearn proporciona una excelente biblioteca para aprender sobre Random foresting. También es una herramienta útil para aprender sobre el análisis y la comparación de características.

```
randForestModel = RandomForestRegressor(n_estimators=100, random_state=8)
crossValMean = np.sqrt(-cross_val_score(estimator=randForestModel,
                                         X=train, y=np.ravel(train_target), cv=10, scoring = scorer)).mean()
crossValSTD = np.sqrt(-cross_val_score(estimator=randForestModel,
                                       X=train, y=np.ravel(train_target), cv=10, scoring = scorer)).std()
```

```
randForestModel.fit(train, np.ravel(train_target))
predictions = randForestModel.predict(test)
predictions = predictions.astype(int)

submission = pd.DataFrame(predictions, columns=["total_cases"])
```

```
submission.insert(0, 'city', test_city)
submission.insert(1, 'year', test_year)
submission.insert(2, 'weekofyear', test_weekofyear)

submission.reset_index()
```

Exportamos a un formato csv los datos de salida para poder enviarlos a ser evaluados en la competición

SUBMISSIONS

| Score | Submitted by | Timestamp |
|---------|--------------|-------------------------|
| 26.6779 | danieledu | 2020-02-09 19:45:18 UTC |
| 26.7692 | danieledu | 2020-02-02 16:38:11 UTC |

Prophet

A principios de 2017 Facebook abrió Prophet, una herramienta de predicción disponible en Python y R. Prophet es particularmente buena para pronosticar series temporales con una fuerte estacionalidad, cambios de tendencias históricas y tendencias que siguen curvas de crecimiento no lineales. Otras buenas razones para utilizar Prophet son la excelente documentación proporcionada y la facilidad general de uso.

Ya que vamos a tratar de hacer predicciones basadas puramente en datos de tiempo, podemos cortar **city**, **weekofyear** y **total_cases** de nuestros datos.

```
train['total_cases'] = labels['total_cases']
prophet_data = train[['city', 'week_start_date', 'total_cases']]
```

```
prophet_data_sj = prophet_data[prophet_data['city'] == 'sj'].drop('city', axis=1)
prophet_data_iq = prophet_data[prophet_data['city'] == 'iq'].drop('city', axis=1)

test_sj = test[test['city'] == 'sj']['week_start_date']
test_iq = test[test['city'] == 'iq']['week_start_date']
```

Para que Prophet funcione correctamente, es obligatorio nombrar la característica de tiempo **ds** y los valores para predecir **y**. Hágalo para ambos dataframes.

```
prophet_data_sj.columns, prophet_data_iq.columns = ['ds', 'y'], ['ds', 'y']
```

Prophet magic

Podemos empezar a usar la librería de fbprophet para pronosticar los valores de nuestros datos de prueba. Para mayor comodidad, crearemos una función que tome los datos del tren, los datos de prueba y dos parámetros de previsión como entrada. El primer paso para establecer una predicción de Prophet es crear una nueva instancia de la clase Prophet. Se puede establecer un montón de parámetros (por ejemplo, estacionalidad_semanal o estacionalidad_diaria) durante la iniciación de esta clase, pero sólo estableceremos la **escala_previa_de_puntos_de_cambio** (por defecto 0,05) y la **estacionalidad_anual** (por defecto 10)

```
def create_forecast(train_data, test_data, flex, seas):

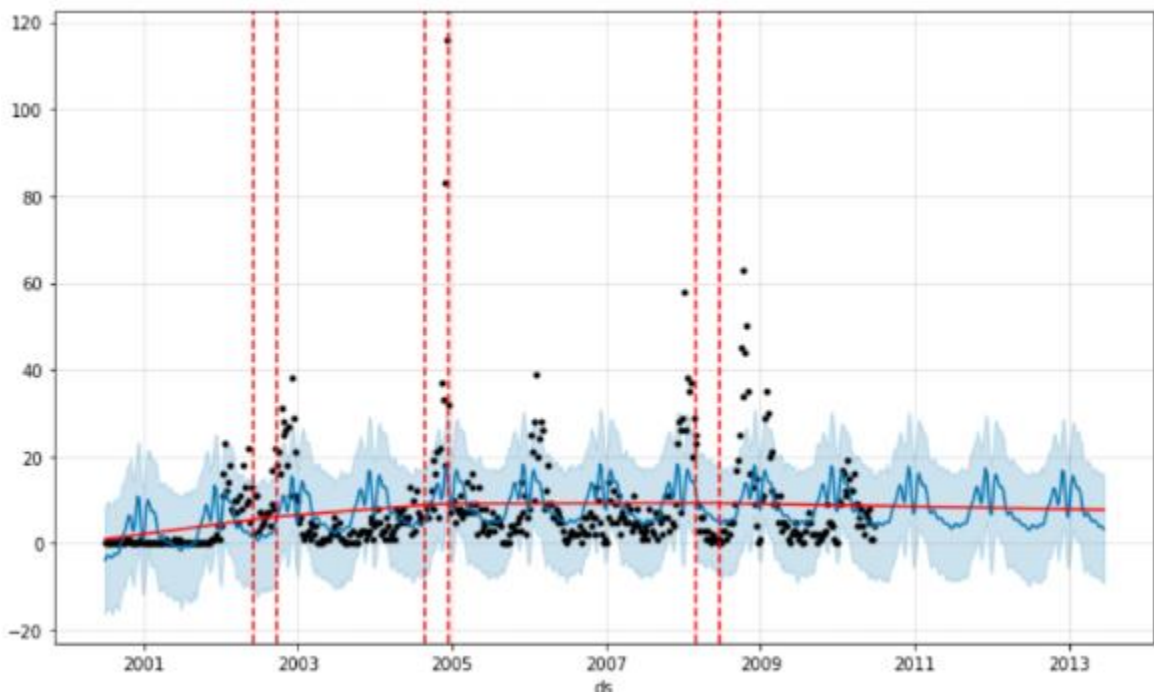
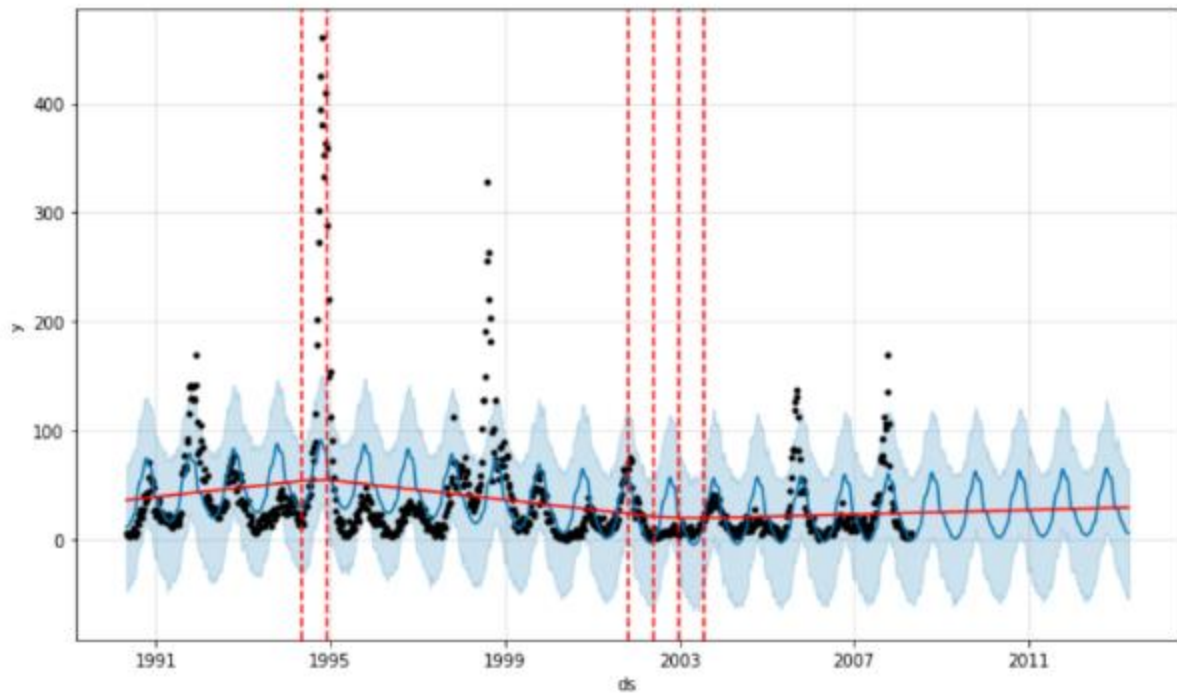
    prophet = Prophet(changepoint_prior_scale=flex, yearly_seasonality=seas)

    prophet.fit(train_data)

    future_frame = prophet.make_future_dataframe(
        periods=len(test_data),
        freq='W')

    forecast = prophet.predict(future_frame)
    prophet.plot(forecast)
    forecast = forecast[['ds', 'yhat']].yhat.apply(lambda x : int(x))

    return forecast[len(train_data):]
```




Mirando los pronósticos, podemos ver que de hecho creamos dos series de pandas con predicciones para nuestros datos de prueba. Junto con la serie, también debería aparecer un gráfico con los datos originales dispersos y las predicciones como una línea continua.

Exportamos a un formato csv los datos de salida para poder enviarlos a ser evaluados en la competición

SUBMISSIONS

| Score | Submitted by | Timestamp | |
|---------|--------------|-------------------------|--|
| 26.6779 | danieledu | 2020-02-09 19:45:18 UTC | |
| 26.7692 | danieledu | 2020-02-02 16:38:11 UTC | |

Usuario de DrivenData:



danieledu

2

ENTRIES

1

AVG #ENTRIES

0

VICTORIES