Análisis y Modelado de la Obesidad mediante Técnicas de Clústeres y Predicción Supervisada

Jheison Stiven Torres Castiblanco

Jonathan Steven Alonso Pinzón

Universidad EAN

Especialización Machine Learning

Grupo MLRU2 - M7V - Virtual - 2024

Carlos Isaac Zainea Maya

14-11-2024

Resumen

Este proyecto se enfoca en el análisis de datos sobre los niveles de obesidad en individuos de los países de México, Perú y Colombia, con base en sus hábitos alimentarios y condición física, por lo que se propone aplicar técnicas de aprendizaje no supervisado y supervisado para mejorar la predicción de los niveles de obesidad. Los objetivos principales incluyeron la identificación de patrones ocultos mediante el uso de clústeres y la evaluación del impacto de dichos patrones en el rendimiento de los modelos de predicción. Los hallazgos clave demostraron que el uso de clústeres como variables adicionales en el modelo mejora la precisión y la capacidad de interpretación.

Introducción

Basados en los perjuicios que genera la obesidad a nivel mundial ha impulsado la necesidad de generar atreves de modelos predictivos poder identificar factores que contribuyen a este fenómeno. Por lo que se plantea explorar los datos mediante técnicas de clústeres y evaluar cómo la inclusión de estos patrones en un modelo de aprendizaje supervisado y no supervisados puede mejorar la predicción de los niveles de obesidad.

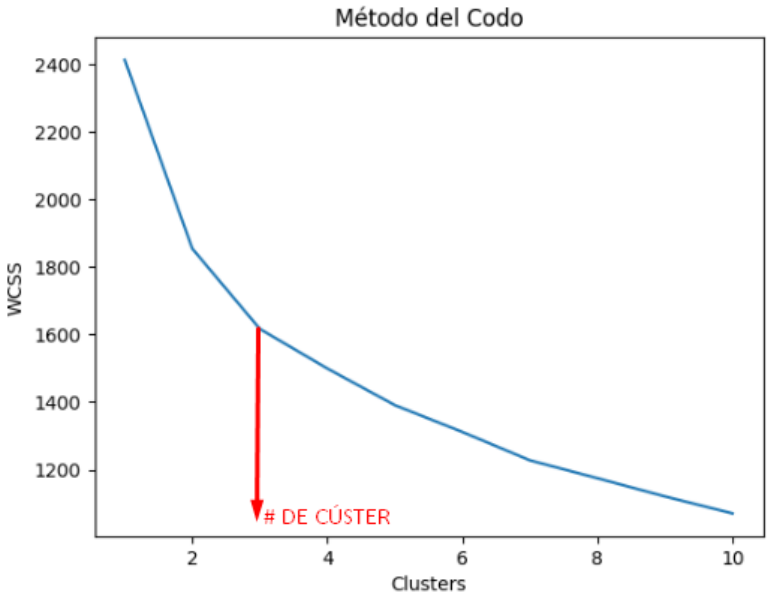
Metodología

El proyecto se desarrolló en varias fases:

1. **Exploración No Supervisada (Clústeres)**:
   - Se utilizó el algoritmo K-Means para segmentar los datos en grupos homogéneos basados en variables contenidas en el data frame excluyendo peso y altura.

Clasificación Modelo NO Supervisado

```
# Elección del número de clústeres (k)
# Utilizando el método del codo para encontrar el número óptimo de clústeres
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Graficar los resultados del método del codo
plt.plot(range(1, 11), wcss)
plt.title('Método del Codo')
plt.xlabel('Clusters')
plt.ylabel('WCSS') # Suma de cuadrados dentro del clúster
plt.show()
```



   - Se evaluó el número óptimo de clústeres mediante el método del codo

```
In [45]: # numero de clusters
         optimal_k = 3

         # entrena Modelado
         kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=10, random_state=0)
         clusters = kmeans.fit_predict(X_scaled)

         cluster_labels = kmeans.labels_

         df['Cluster'] = cluster_labels
```

```
In [46]: df.head()
```

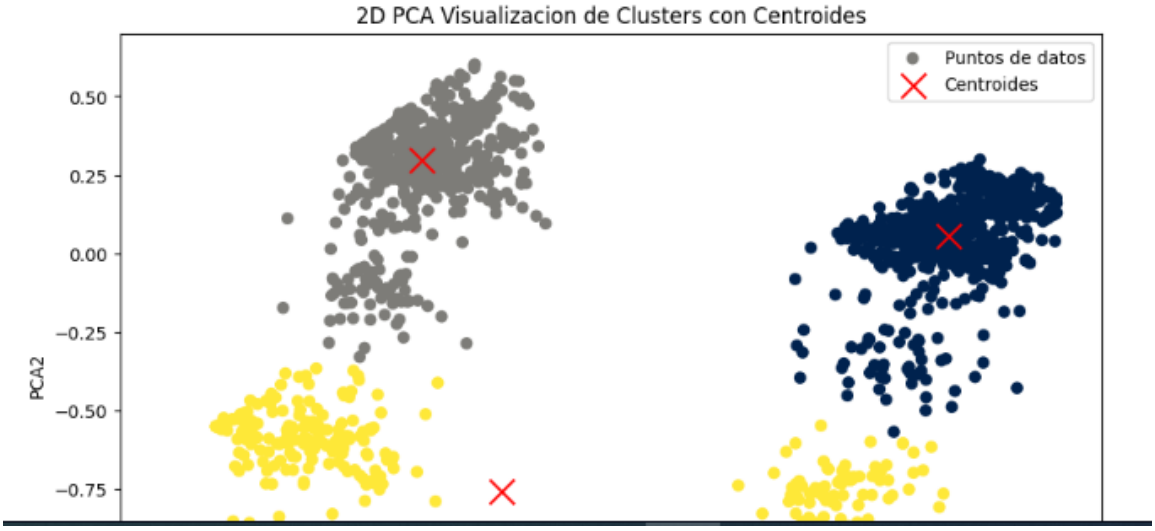
Out[46]:

ght	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad	Cluster
4.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight	1
8.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight	1
7.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight	0
7.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I	2
9.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II	2

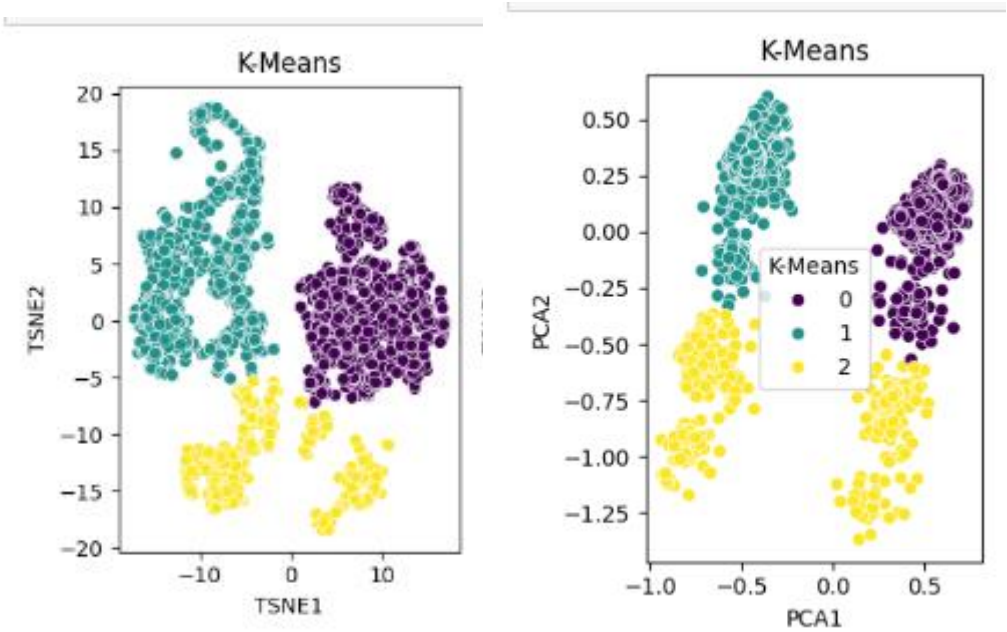
#De cluster

- Se realiza análisis de componentes Principales (PCA) para Reducir la Dimensionalidad de Datos donde se seleccionan 3 componente.

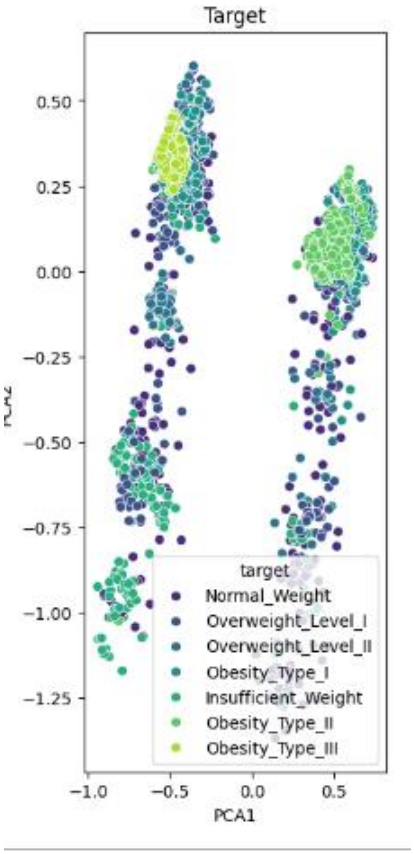
```
from sklearn.decomposition import PCA

# inicializa PCA
pca = PCA(n_components=3)  # De componentes selecionados
df_pca = pca.fit_transform(X_scaled)
centroids = kmeans.cluster_centers_
centroids_pca = pca.transform(centroids)

# PCA inicializacion y transformacion
plt.figure(figsize=(10, 7))
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=clusters, cmap='cividis', marker='o', label='Puntos de datos')
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], c='red', marker='X', s=200, label='Centroides')
plt.title('2D PCA Visualizacion de Clusters con Centroides')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.legend()
plt.show()
```

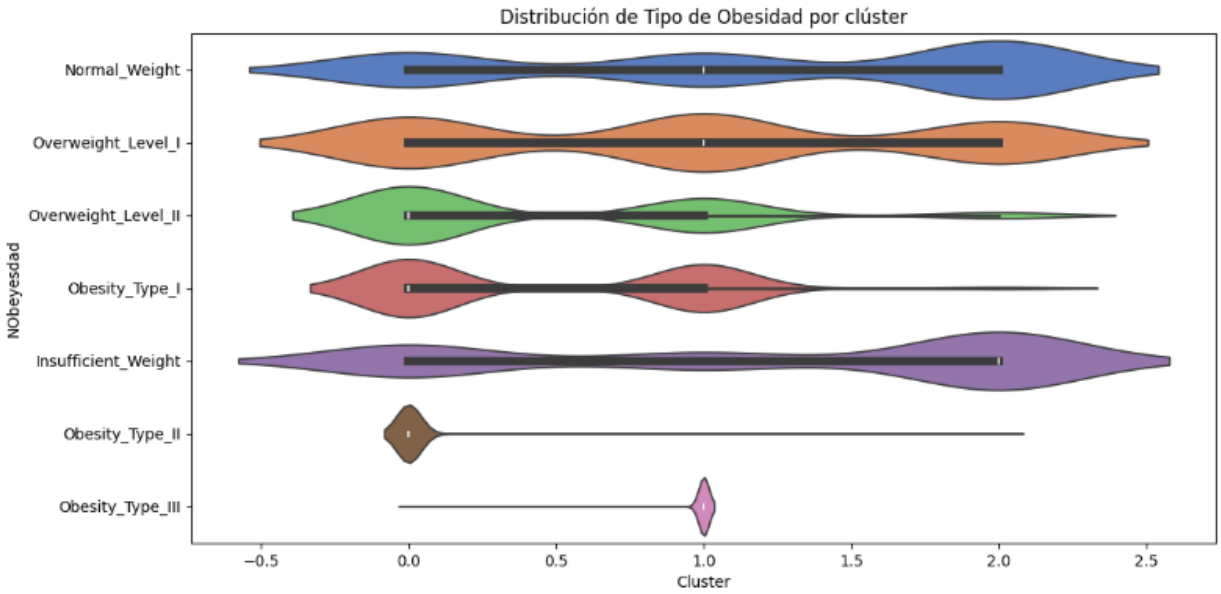


- Se valida visualización de resultado K-Means aplicando TSNE y PCA



- Variable objetivo distribuida por clúster donde se evidencia la distribución de los tipos de obesidad. El tipo de obesidad II se concentra en el clúster 0





2. **Entrenamiento Supervisado**:

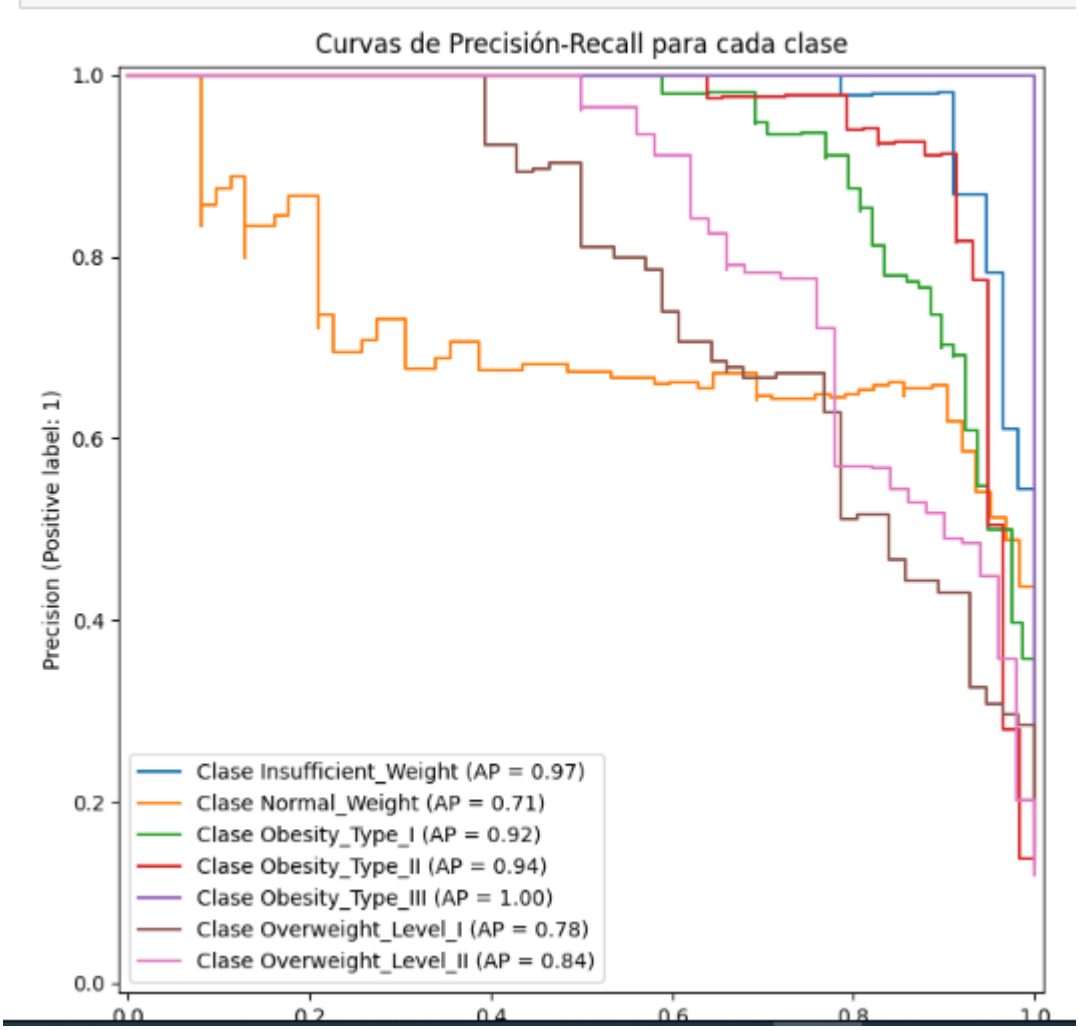
- Se entrenaron modelos de clasificación, como Random Forest Y regresión logistica, tanto con las variables originales como incluyendo los clústeres como una nueva variable.
- Se evidencia rendimiento similar con o sin incluir la variable clúster por lo que se incluye en el modelo.

```
# Entrenamos el modelo
modelo_rf = RandomForestClassifier(n_estimators=100, random_state=42)
modelo_rf.fit(X_train_rf, y_train_rf)

## Predicción
y_pred_rf = modelo_rf.predict(X_test_rf)

## Evaluación
print(classification_report(y_test_rf, y_pred_rf))
```

```
                     precision    recall  f1-score   support

Insufficient_Weight       0.88      0.95      0.91        56
      Normal_Weight       0.68      0.73      0.70        62
     Obesity_Type_I       0.85      0.85      0.85        78
    Obesity_Type_II       0.83      0.95      0.89        58
   Obesity_Type_III       1.00      1.00      1.00        63
 Overweight_Level_I       0.84      0.73      0.78        56
Overweight_Level_II       0.80      0.66      0.73        50

           accuracy                           0.84       423
          macro avg       0.84      0.84      0.84       423
       weighted avg       0.84      0.84      0.84       423
```

mejor rendiemiento

- El modelo seleccionado es el de Random Forest ya que alcanza una precisión global del 84% identifica con mayor precisión los casos de pesos insuficiente como de obesidad y mantiene un balance sólido entre precisión, recall y f1-score en todas las clases, reflejando una mejor capacidad de generalización y consistencia en las predicciones.



- Se optimizaron los hiperparámetros mediante búsqueda en cuadrícula (Grid Search) para maximizar la precisión.

Aplicando Gridsearch

```
[5]: from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score, classification_report
     from scipy.stats import randint

     # Paso 1: Preparar los datos
     X = pre1
     y = df['NObeyesdad']  # Variable objetivo

     # Dividir los datos en entrenamiento y prueba
     X_train_gs, X_test_gs, y_train_gs, y_test_gs = train_test_split(X, y, test_size=0.2, random_state=42)

     # Paso 2: Definir el modelo
     rf = RandomForestClassifier(random_state=42)

     # Paso 3: Configurar el espacio de búsqueda de hiperparámetros
     # Para GridSearchCV
     param_grid = {
         'n_estimators': [50, 100, 200],
         'max_depth': [None, 10, 20, 30],
         'min_samples_split': [2, 5, 10]
     }

     # Paso 4: Ejecutar la búsqueda de hiperparámetros

     # Grid Search
     grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
     grid_search.fit(X_train_gs, y_train_gs)
     print("Mejores parámetros (Grid Search):", grid_search.best_params_)

     # Paso 5: Evaluar el modelo con los mejores hiperparámetros en el conjunto de prueba
     best_rf_grid = grid_search.best_estimator_

     # Predicciones y evaluación
     y_pred_grid = best_rf_grid.predict(X_test_gs)

     print("Rendimiento de Grid Search:")
     print(classification_report(y_test_gs, y_pred_grid))
```

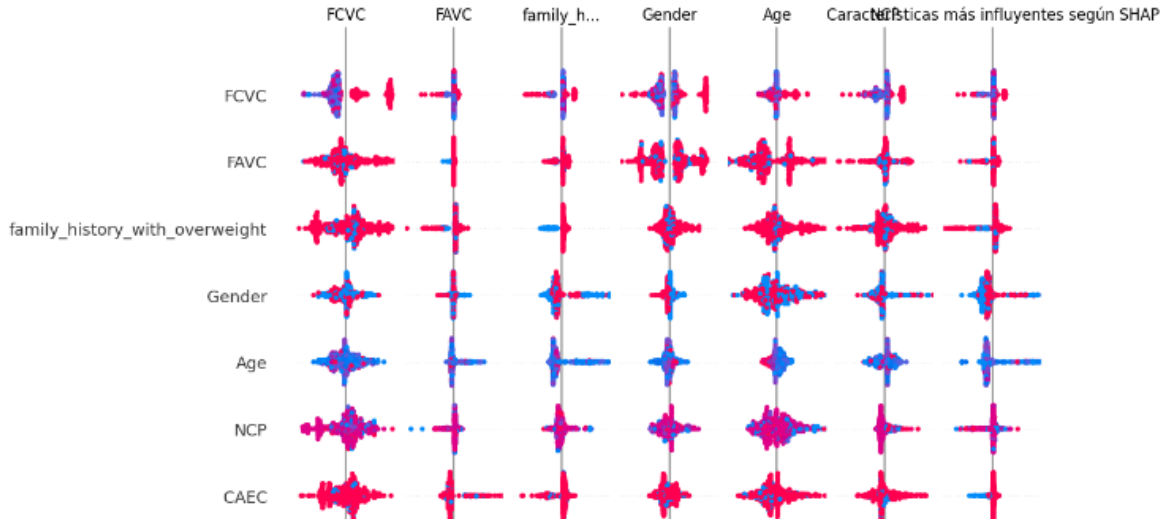
```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Mejores parámetros (Grid Search): {'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 200}
Rendimiento de Grid Search:
                     precision    recall  f1-score   support

Insufficient_Weight       0.92      0.96      0.94        56
      Normal_Weight       0.66      0.76      0.71        62
     Obesity_Type_I       0.87      0.83      0.85        78
    Obesity_Type_II       0.81      0.95      0.87        58
   Obesity_Type_III       1.00      1.00      1.00        63
 Overweight_Level_I       0.83      0.71      0.77        56
Overweight_Level_II       0.85      0.66      0.74        50

           accuracy                           0.84       423
          macro avg       0.85      0.84      0.84       423
       weighted avg       0.85      0.84      0.84       423
```

3. **Evaluación del Rendimiento**:

- Se aplicaron técnicas de interpretabilidad como SHAP para entender la influencia de las variables en las predicciones.

```
import shap

explainer = shap.Explainer(best_rf_grid)
shap_values = explainer.shap_values(X_test_gs)
#shap_values = explainer(X_test)

#shap.summary_plot(shap_values, X_test,show=False)
plt.figure(figsize=(10, 8))
shap.summary_plot(shap_values, X_test_gs, plot_type="bar", show=False)
plt.title("Características más influyentes según SHAP")
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```



Resultados

Los resultados mostraron que la inclusión de clústeres mejoró de forma mínima el rendimiento de los modelos. Las visualizaciones revelaron que los clústeres capturaban patrones complejos relacionados con la obesidad y otros factores. Las técnicas de interpretabilidad, como los gráficos SHAP, ayudaron a destacar las variables más influyentes en las predicciones, confirmando la importancia de considerar las agrupaciones en la modelación.

Conclusiones y Recomendaciones

La inclusión de clústeres en el proceso de modelado podría ser una estrategia eficaz para mejorar la predicción de los niveles de obesidad. Esto sugiere que futuros trabajos deberían explorar otras técnicas de segmentación y considerar métodos de interpretabilidad para garantizar modelos más transparentes y robustos.

Referencias

- Estimation of Obesity Levels Based On Eating Habits and Physical Condition

  https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

- Bagnato, J. (2017). Aprende Machine Learning en Español Teoría + Práctica.

  https://leanpub.com/aprendemlEnlaces a un sitio externo.