Introducción a la Ciencia de Datos

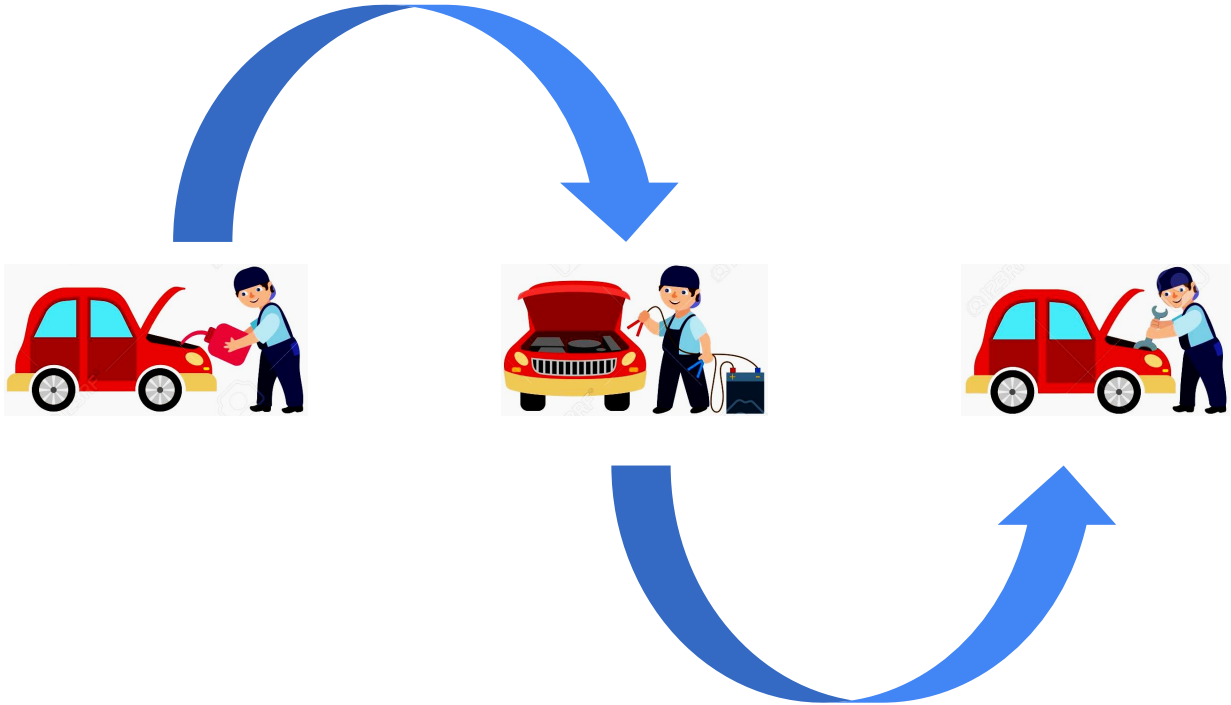
Maestría en Ciencias de la Computación

Dr. Irvin Hussein López Nava

05

Meta-aprendizaje

How to troubleshoot car problems



5.2 Boosting

Centrarse en el boost

- En los **métodos secuenciales**, los diferentes modelos débiles combinados no se ajustan de forma independiente unos de otros.
- La idea es ajustar los modelos de forma **iterativa**, de modo que el entrenamiento del modelo en un paso determinado dependa de los modelos ajustados en los pasos anteriores.
- El *boosting* produce un modelo de ensamble que, en general, está menos **sesgado** que los aprendices débiles que lo componen.

Boosting vs Bagging

- **Ambos métodos** funcionan con el mismo espíritu:
  - Construyen un conjunto de modelos agregados para obtener un aprendiz fuerte que obtenga mejores resultados.
- Sin embargo, a diferencia del **bagging**, cuyo principal objetivo es reducir la varianza, el **boosting** consiste en ajustar secuencialmente múltiples aprendices débiles de forma adaptativa:
  - Cada modelo se ajusta dando más importancia a las observaciones del conjunto de datos que no fueron bien “tratadas” por los modelos anteriores en la secuencia.
- Al final del proceso se obtiene un aprendiz fuerte con un sesgo menor.

Modelos base para Boosting

- Al centrarse en la reducción del **sesgo**, los modelos base que se consideran para *boosting* son aquellos con baja varianza pero alto sesgo,
  - e.g., árboles de decisión poco profundos.
- Otra razón para usar este tipo de modelos en *boosting* es que, en general, su ajuste es menos costoso desde el punto de vista computacional.
  - Dado que los cálculos no pueden realizarse en paralelo (a diferencia de *bagging*), podría resultar demasiado costoso ajustar secuencialmente varios modelos complejos.
- Las preguntas: ¿cómo se ajustarán secuencialmente los modelos? y ¿cómo se agregarán tales modelos?

Adaptive Boosting

- En el *boosting* adaptativo, también llamado **adaboost**, un modelo de ensamble se define como una suma ponderada de *L* aprendices débiles:

$$s_L(x) = \sum_{l=1}^{L} c_l w_l(x)$$

  donde *c* son los coeficientes y *w* son los modelos base con *x* como entrada.
- Encontrar el mejor modelo de ensamble es un problema de optimización.
- En lugar de tratar de encontrar todos los coeficientes y aprendices débiles que den el mejor modelo aditivo global, se utiliza un proceso de optimización iterativo, aunque pueda conducir a una solución subóptima.

Adaptive boosting

- Se añaden aprendices débiles uno a uno, buscando en cada iteración el mejor par posible (coeficiente - aprendiz débil) para añadir al modelo de ensamble actual.

$$s_l(x) = s_{l-1}(x) + c_l w_l(x)$$

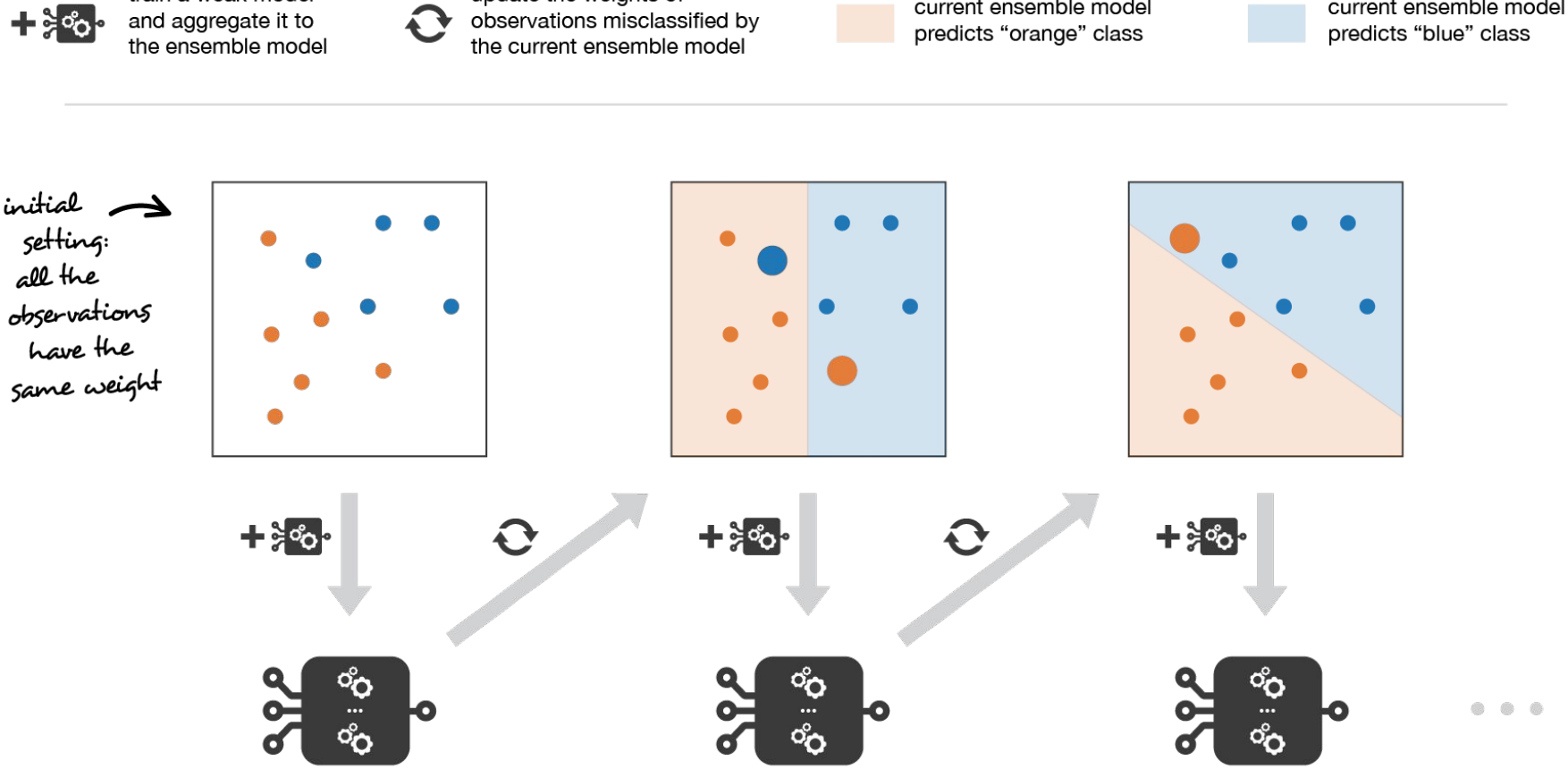
donde c_l y w_l se eligen de tal forma que s_l es el modelo que mejor se ajusta a los datos de entrenamiento y es la mejor posible mejora sobre s_{l-1}.

$$(c_l, w_l(x)) = arg\ \underset{c_l, w_l(x)}{min} \sum_{n=1}^{N} e(y_n, s_{l-1}(x_n) + cw(x_n))$$

donde *e* es la función de pérdida/error.

Adaptive boosting

- En lugar de optimizar globalmente sobre todos los modelos *L* de la suma, se aproxima el óptimo localmente añadiendo uno a uno los aprendices débiles a un modelo fuerte.
- Cuando se considera una clasificación binaria, el algoritmo **adaboost** puede reescribirse en un proceso como el siguiente:
    1. Actualizar los pesos de las observaciones en el conjunto de datos y entrenar un nuevo aprendiz débil con especial atención a las observaciones mal clasificadas por el modelo de ensamble actual.
    2. Añadir el aprendiz débil a la suma ponderada de acuerdo a un coeficiente de actualización que exprese el rendimiento de este modelo débil: cuanto mejor sea el rendimiento de un aprendiz débil, más contribuirá al aprendiz fuerte.

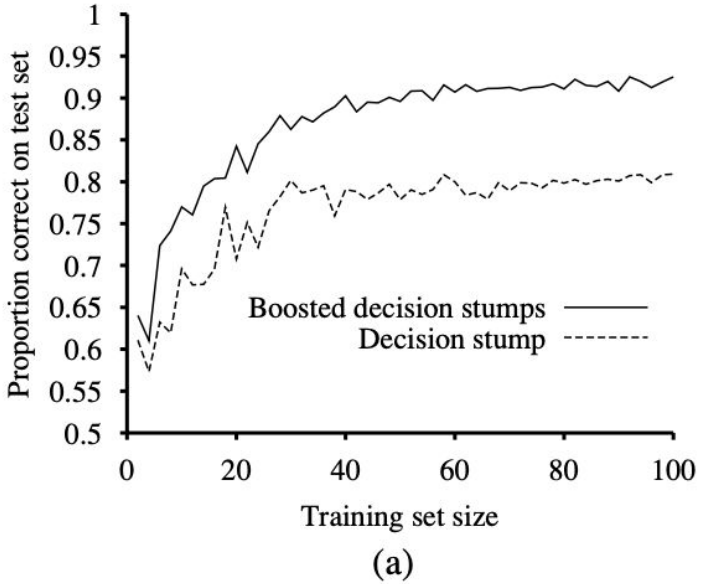

train a weak model and aggregate it to the ensemble model
update the weights of observations misclassified by the current ensemble model
current ensemble model predicts “orange” class
current ensemble model predicts “blue” class
initial setting: all the observations have the same weight

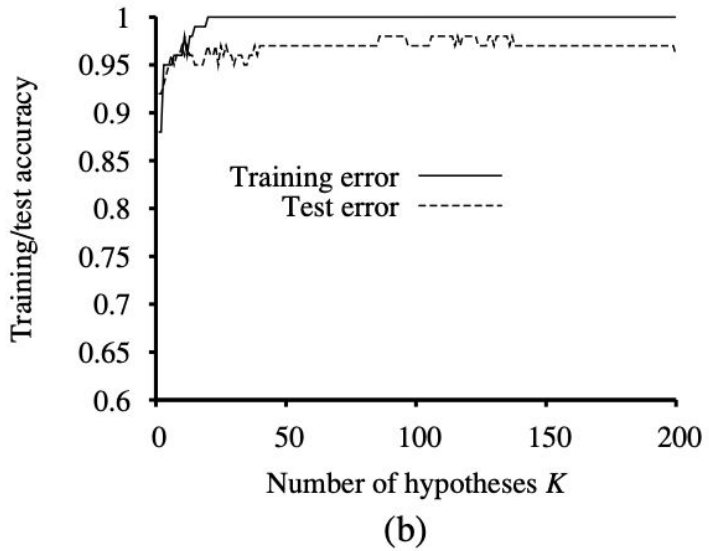
Adaboost algorithm

function ADABOOST($examples, L, K$) **returns** a weighted-majority hypothesis
  inputs: $examples$, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
    L, a learning algorithm
    K, the number of hypotheses in the ensemble
  local variables: **w**, a vector of N example weights, initially $1/N$
    h, a vector of K hypotheses
    z, a vector of K hypothesis weights

  for $k = 1$ **to** K **do**
    $\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$
    $error \leftarrow 0$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$
    $\mathbf{w} \leftarrow$ NORMALIZE(**w**)
    $\mathbf{z}[k] \leftarrow \log(1 - error)/error$
  return WEIGHTED-MAJORITY(**h**, **z**)

Ejemplo de ejecución





Más sobre Adaboost

- *Adaboost* actualiza los pesos de las observaciones en cada iteración.
- Las ponderaciones de las observaciones bien clasificadas disminuyen en relación con las ponderaciones de las observaciones mal clasificadas.
- Los modelos que obtienen mejores resultados tienen ponderaciones más altas en el modelo conjunto final.
- Puede identificar valores atípicos, ya que se centra en ejemplos difíciles de clasificar.
- Demasiados valores atípicos pueden degradar el rendimiento de la clasificación y aumentar drásticamente el tiempo de convergencia.

Gradient boosting

- En *boosting* por gradiente, el modelo de ensamble a construir es también una suma ponderada de aprendices débiles.

$$s_L(x) = \sum_{l=1}^{L} c_l w_l(x)$$

- Como en el caso de adaboost, encontrar el modelo óptimo de esta forma es demasiado difícil y se requiere un enfoque iterativo.
- La principal diferencia con el *boosting* adaptativo es la definición del proceso de optimización secuencial.

Gradient boosting

- Gradient boosting convierte el problema en uno de descenso de gradiente: en cada iteración se ajusta un aprendiz débil al gradiente opuesto del error de ajuste actual con respecto al modelo de ensamble actual.
- El proceso de descenso de gradiente sobre el modelo de ensamble puede escribirse:

$$s_l(x) = s_{l-1}(x) - c_l \nabla_{s_{l-1}} E(s_{l-1})(x)$$

donde *E* es el error de ajuste del modelo dado y

$$-\nabla_{s_{l-1}} E(s_{l-1})(x)$$

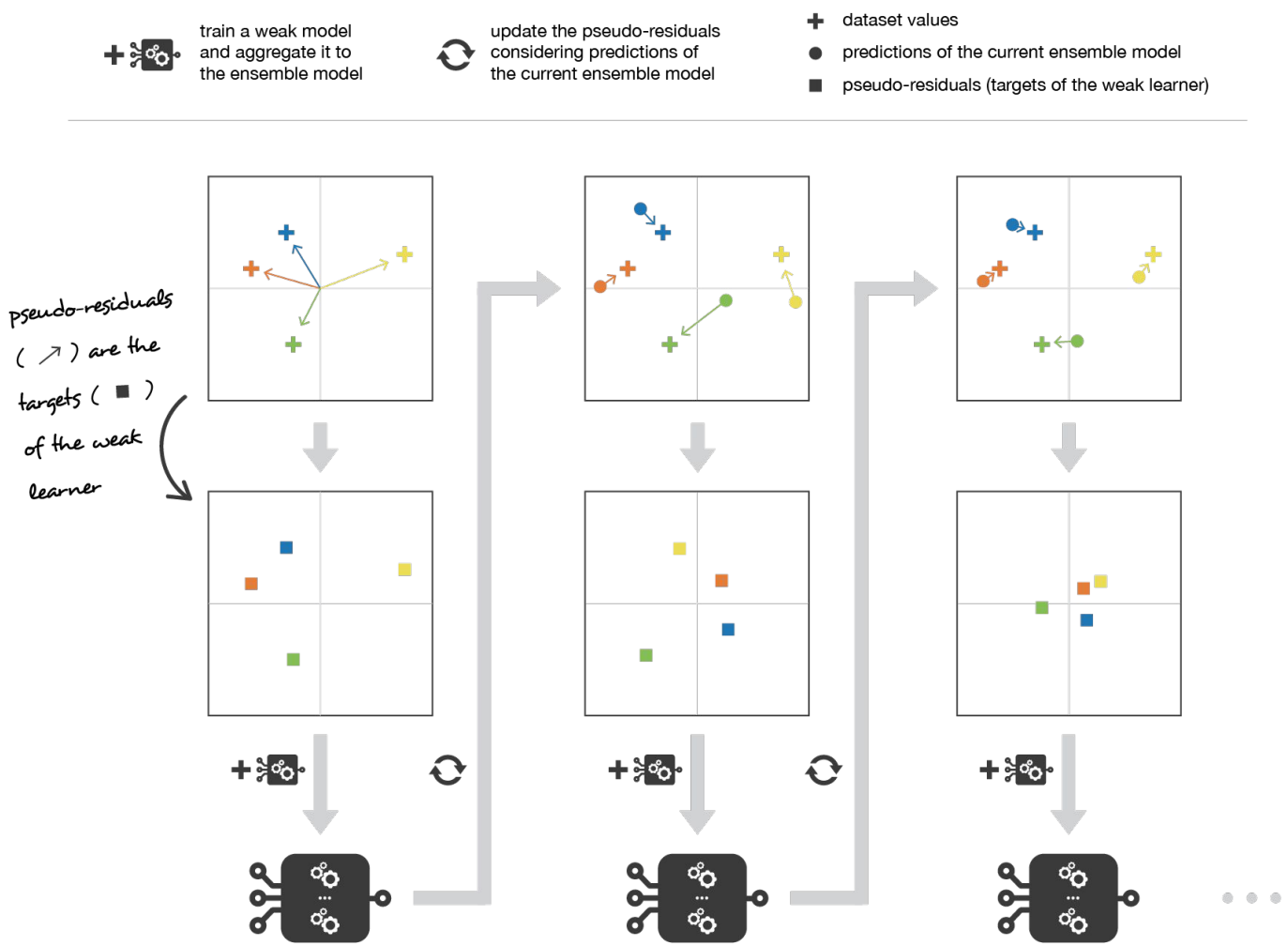
es el gradiente opuesto del error de ajuste en el paso *l–1*.

Gradient boosting

- Lo opuesto al gradiente es una función para las observaciones del conjunto de datos de entrenamiento: estas evaluaciones se denominan **pseudo-residuales** adjuntas a cada observación.
- Se ajusta un aprendiz débil a los pseudo-residuos calculados para cada observación.
- Por último, el coeficiente c se calcula siguiendo un proceso de optimización unidimensional (búsqueda lineal para obtener el mejor tamaño de paso c).

Gradient boosting

- Al inicio del algoritmo, los pseudo-residuos se igualan a los valores de observación.
- A continuación, se repite L veces (modelos de la secuencia):
  - Ajustar el mejor aprendiz débil posible a los pseudo-residuos.
  - Calcular el valor del tamaño de paso óptimo: actualizar el modelo de ensamble en la dirección del nuevo aprendiz débil.
  - Actualizar el modelo de ensamble añadiendo el nuevo aprendiz débil multiplicado por el tamaño del paso.
  - Calcular los nuevos pseudo-residuos: dirección de actualización de las predicciones del modelo de ensamble.


train a weak model and aggregate it to the ensemble model
update the pseudo-residuals considering predictions of the current ensemble model
dataset values
predictions of the current ensemble model
pseudo-residuals (targets of the weak learner)
pseudo-residuals (↗) are the targets (■) of the weak learner

AdaBoost vs Gradient Boosting

- El *boosting* adaptativo intenta resolver en cada iteración exactamente el problema de optimización **local** (encontrar el mejor aprendiz débil y su coeficiente para añadirlo al modelo fuerte),
- En cambio, el *gradient boosting* utiliza un enfoque de descenso de gradiente y puede adaptarse más fácilmente a un gran número de funciones de pérdida.
- Así, el *gradient boosting* puede considerarse una generalización de *adaboost* para funciones de pérdida diferenciables arbitrarias.

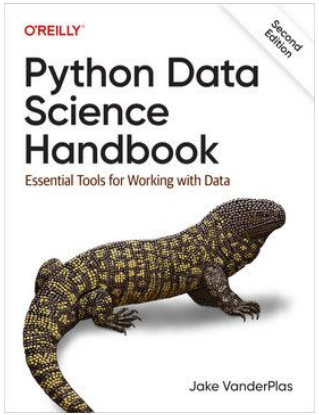
Variantes de Adaboost

- *Real AdaBoost*: el valor de probabilidad se mapea mediante una función logarítmica, y el último clasificador es el final de todas las funciones de mapeo.
- *Gentle AdaBoost*: en cada iteración, hace una regresión ponderada basada en los mínimos cuadrados, y el último clasificador de todas las funciones de regresión.
- *LogitBoost*: es parecido al adaboost suave, pero la variable *z* se actualiza constantemente cada vez que se ajusta la regresión.

Resumen de Boosting

- Combinar clasificadores débiles para obtener un clasificador muy fuerte
  - **Clasificador débil** – ligeramente mejor que el aleatorio en los datos de entrenamiento.
  - **Clasificador resultante muy potente**: puede llegar a proporcionar un error de entrenamiento cero.
- *Boosting* vs regresión logística
  - Optimización única (LR) frente a mejora incremental de la clasificación (B)
- La aplicación más popular de Boosting:
  - Decision stump boosted!
  - Muy sencillo de aplicar, clasificador muy eficaz.

Extra Libro



GitHub

- ?

Gracias!

¿Alguna pregunta?

hussein@cicese.mx
https://sites.google.com/view/husseinlopeznava

SCAN ME

CREDITS: This presentation was based on a template by **Slidesgo**, and includes icons by **Flaticon**.