UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

# Documentazione Progetto Ingegneria della Conoscenza 21/22 “Analisi e previsioni dei prezzi di biglietti aerei”

## Gruppo di lavoro


- Raffaele Distante, 669014, r.distante@studenti.uniba.it
- Filippo Di Gesù, 663245, f.digesu1@studenti.uniba.it


## AA 2021-22 (ICon8)

LINK A REPOSITORY: https://github.com/filippodiges/Progetto-Icon/

# Introduzione

Il presente caso di studio è stato sviluppato attraverso il linguaggio Python nell'applicazione web "Jupyter Notebook" che permette di creare e condividere documenti contenenti codice sorgente, testo descrittivo, immagini, grafici e altro ancora.

Abbiamo trattato il tema dell'analisi e la previsione dei prezzi di biglietti aerei utilizzando dati appartenenti al dataset "Flight Fare Prediction" presente sulla piattaforma Kaggle. Il dataset fa riferimento ai dati nel solo 2019 delle compagnie aeree presenti, con un numero di elementi pari a 10683. Nel dataset sono descritti gli elementi associati ad un volo tra cui: Airline, Date_of_Journey, Source, Destination, Route, Dep_Time, Arrival_Time, Duration, Total_Stops, Additional_Info, Price.

Dall'analisi esplorativa e il preprocessing abbiamo analizzato e processato i dati per il training set e il test set, dando un peso specifico alle feature più importanti per questa fase.

È stato, infine, affrontata la sezione di addestramento dei vari modelli di apprendimento supervisionato.

# Sommario

Quando si affrontano problemi di Machine Learning, esistono generalmente due tipi di dati (e di modelli di apprendimento automatico): Dati supervisionati: hanno sempre uno o più target associati. Dati non supervisionati: non hanno alcuna variabile target. Un problema supervisionato è molto più semplice da affrontare rispetto ad uno non supervisionato. Un problema in cui si richiede di prevedere un valore è noto come problema supervisionato. Ad esempio, se il problema è quello di prevedere i prezzi delle case, dati i prezzi storici delle case, con caratteristiche come la presenza di un ospedale, di una scuola o di un supermercato, la distanza dal più vicino mezzo di trasporto pubblico, ecc. Allo stesso modo, se ci vengono fornite immagini di cani e gatti e sappiamo in anticipo quali sono i gatti e quali i cani, e se il compito è quello di creare un modello che preveda se un'immagine fornita è di un gatto o di un cane, il problema è considerato supervisionato.

Nel Dataset in questione, abbiamo un problema di apprendimento automatico supervisionato, per la previsione del prezzo dei voli aerei.

I prezzi dei biglietti aerei possono essere difficili da indovinare: oggi potremmo vedere un prezzo, domani il prezzo dello stesso volo sarà diverso. Come data scientists, dimostreremo che, con i dati giusti, tutto può essere previsto. Qui vengono forniti i prezzi dei biglietti aerei di varie compagnie aeree tra i mesi di marzo e giugno 2019 e tra varie città.

Il nostro dataset ha 10683 righe e 13 colonne con valori duplicati. La predizione è stata eseguita senza valori duplicati.

Poiché la maggior parte delle colonne è costituita da dati numerici, per questo motivo è stato utilizzato un regressore per una migliore previsione e tra tutti i modelli di regressore CatBoost (punteggio r2: 87,54%, con Hyperparameter Tuning) ha ottenuto risultati migliori rispetto ad altri modelli, in particolare rispetto a XGBoost, Decision Tree e Random Forest.

# Elenco argomenti di interesse

**Questo progetto è trattato nelle seguenti fasi:**

## Fase 1: Importare le librerie e scaricare i dati

## Fase 2: Analisi esplorativa dei dati (EDA) e Visualizzazione Esplorazione dei dati

- Pulizia dei dati
- Percentuale di valori mancanti
- Identificazione e eliminazione dei valori NaN/Null
- Identificazione e eliminazione dei valori duplicati
- Preelaborazione dei dati
- Estrazione della data e dell'ora
- Visualizzazione
- Rapporti automatizzati con AutoViz

## Fase 3: Preparazione del dataset per l'addestramento

- Identificare le colonne numeriche
- Identificare le colonne categoriche
- Identificare le colonne di input
- Identificare le colonne target
- Grafico di correlazione per le colonne di input
- Processo di trasformazione delle caratteristiche
- Scala dei valori numerici con MinMaxScaler
- Codifica delle colonne categoriali con OneHotEncoding
- Addestrare la divisione dei test con 80-20

## Fase 4: Costruzione del modello di tutti i regressori e valutazione delle prestazioni

- **Costruire tutti i modelli di regressione**
    - Logistic Regression
    - Linear Regression
    - Ridge
    - Lasso
    - Decision Tree
    - Random Forest
    - K Nearest Neighbors (KNN)
    - Naive Bayes (NB)
    - Support Vector Regressor (SVR)
    - Gradient Boosting
    - XGBoost
    - XGBRF
    - LightGBM
    - CatBoost
    - AdaBoost

    - Bagging
    - Stochastic Gradient Descent (SGD)

- **Trovare il modello migliore** (CatBoost r2 score: 0.8754)
- Valori effettivi e previsti
- Feature Importance
- Hyperameter Tuning per il miglioramento del modello
- Valutazione del modello con il punteggio r2, i valori MAE, MSE e RMSE
- Metriche e punteggi: Quantificare la qualità delle previsioni
- Dare un nuovo ingresso e prevedere i valori target della macchina

## Analisi esplorativa dei dati (EDA) e Visualizzazione

I risultati della fase EDA sono i seguenti:

- Comprensione del dataset e aiuto nella pulizia del dataset.
- Fornisce un quadro chiaro delle caratteristiche e delle relazioni tra di esse.
- Fornire linee guida per le variabili essenziali e per rimuovere le variabili non essenziali.
- Gestire i valori mancanti o gli errori umani.
- Identificare gli outlier.
- Consente di massimizzare gli insight di un set di dati.

Questo processo richiede tempo ma è molto efficace.

Dopo il caricamento appunto, è importante controllare le informazioni complete dei dati, perché possono indicare molte informazioni nascoste, come i valori nulli in una colonna o in una riga.

In qualsiasi dataset del mondo reale, ci sono sempre alcuni valori nulli. Non importa se si tratta di un problema di regressione, classificazione o di qualsiasi altro tipo, nessun modello è in grado di gestire questi valori NULL o NaN da solo, quindi è necessario intervenire.

Controllare se ci sono valori nulli o meno. Se sono presenti, si può procedere come segue:

- Imputare i dati utilizzando il metodo di imputazione di Sklearn.
- Riempire i valori NaN con la media, la mediana e la modalità utilizzando il metodo fillna().

## Strumenti utilizzati

Abbiamo sfruttato diversi strumenti per l'analisi esplorativa dei dati e la loro visualizzazione, tra cui:

Pandas, Matplotlib, Seaborn, Plotly, Missingno e AutoViz.

# Preparazione del dataset per l'addestramento

## Sommario

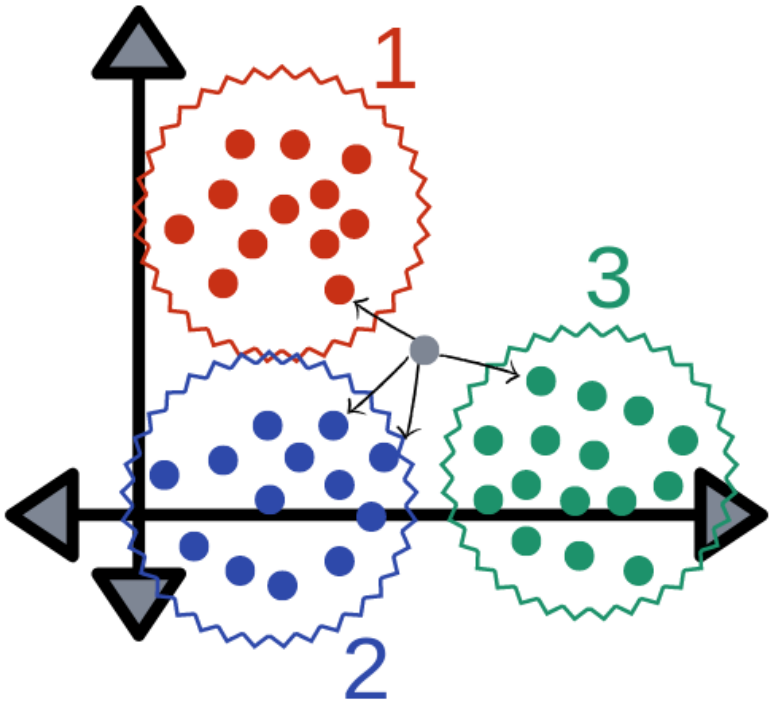
La preelaborazione dei dati è una fase fondamentale del Machine Learning, poiché la qualità dei dati e le informazioni utili che se ne possono ricavare influiscono direttamente sulla capacità di apprendimento del nostro modello; pertanto, è estremamente importante preelaborare i dati prima di inserirli nel modello.
Innanzitutto identifichiamo le colonne di input e la colonna Target. Dopodichè identifichiamo le colonne numeriche e le colonne categoriche del dataset.

# Features Transformation e Scaling - MinMaxScaler, OneHotEncoding

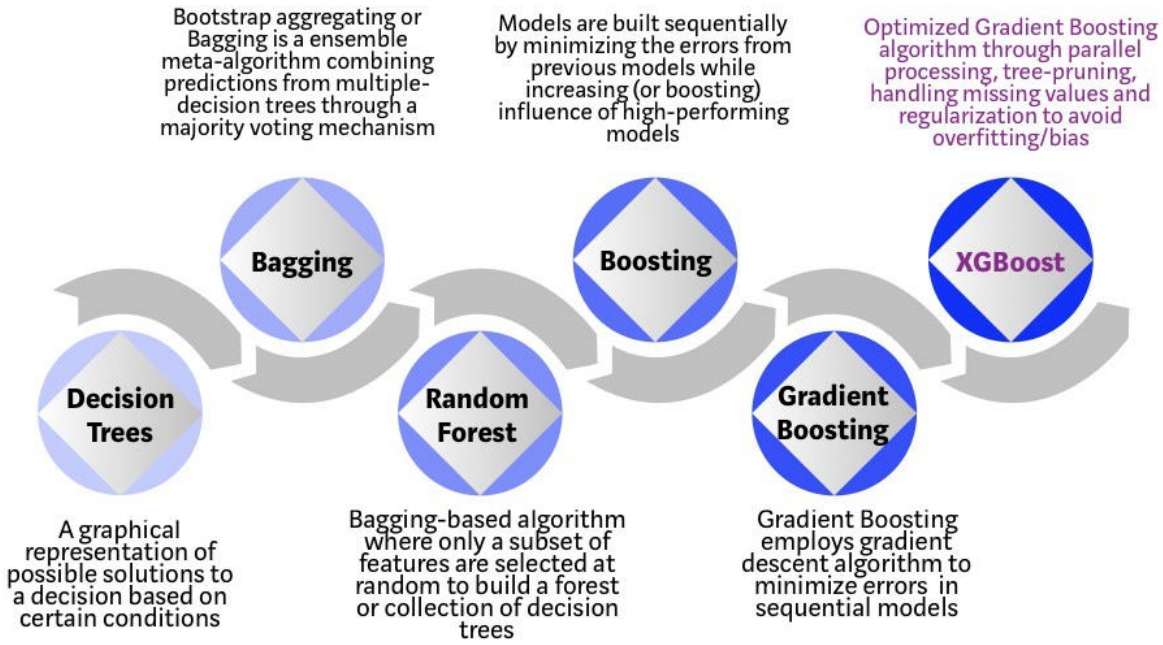
La prima domanda che dobbiamo porci è: perché abbiamo bisogno di scalare le variabili del nostro set di dati?

Alcuni algoritmi di apprendimento automatico sono sensibili alla scalatura delle caratteristiche, mentre altri sono praticamente invarianti.

Algoritmi basati sulla distanza: Gli algoritmi basati sulla distanza, come "KNN", "K-means" e "SVM", sono maggiormente influenzati dalla gamma di caratteristiche. Questo perché dietro le quinte utilizzano le distanze tra i punti di dati per determinarne la somiglianza. Se due caratteristiche hanno scale diverse, è possibile che venga attribuito un peso maggiore alle caratteristiche con un'ampiezza più elevata. Questo avrà un impatto sulle prestazioni dell'algoritmo di apprendimento automatico e, ovviamente, non vogliamo che il nostro algoritmo sia influenzato da una sola caratteristica. Pertanto, prima di impiegare un algoritmo basato sulla distanza, scaliamo i dati in modo che tutte le caratteristiche contribuiscano in egual misura al risultato.



Algoritmi ad albero: Gli algoritmi ad albero, invece, sono abbastanza insensibili alla scala delle caratteristiche. Si pensi che un albero decisionale divide un nodo solo in base a una singola caratteristica. L'albero decisionale divide un nodo in base a una caratteristica che aumenta l'omogeneità del nodo. Questa suddivisione su una caratteristica non è influenzata da altre caratteristiche. Pertanto, l'effetto delle altre caratteristiche sulla divisione è praticamente nullo. Questo li rende invarianti rispetto alla scala delle caratteristiche!




La normalizzazione è una tecnica di scalatura in cui i valori vengono spostati e ridimensionati in modo che risultino compresi tra 0 e 1. È nota anche come scalatura Min-Max.

Ecco la formula per la normalizzazione:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Qui, Xmax e Xmin sono rispettivamente i valori massimo e minimo della caratteristica.

Quando il valore di X è il valore minimo della colonna, il numeratore sarà 0 e quindi X' è 0 D'altra parte, quando il valore di X è il valore massimo della colonna, il numeratore è uguale al denominatore e quindi il valore di X' è 1 Se il valore di X è compreso tra il valore minimo e il valore massimo, allora il valore di X' è compreso tra 0 e 1

La standardizzazione è un'altra tecnica di scalatura in cui i valori vengono centrati intorno alla media con una deviazione standard unitaria. Ciò significa che la media dell'attributo diventa zero e la distribuzione risultante ha una deviazione standard unitaria.

Ecco la formula della standardizzazione:

$$Z = \frac{X - \mu}{\sigma}$$

La normalizzazione è utile quando si sa che la distribuzione dei dati non segue una distribuzione gaussiana. Può essere utile negli algoritmi che non assumono alcuna distribuzione dei dati, come K-Nearest Neighbors e le reti neurali.

La standardizzazione, invece, può essere utile nei casi in cui i dati seguono una distribuzione gaussiana. Tuttavia, questo non deve essere necessariamente vero. Inoltre, a differenza della normalizzazione, la standardizzazione non ha un intervallo di valori limite. Pertanto, anche se i dati presentano dei valori anomali, la standardizzazione non li influenzerà.

Tuttavia, alla fine, la scelta di utilizzare la normalizzazione o la standardizzazione dipende dal problema e dall'algoritmo di apprendimento automatico utilizzato. Non esiste una regola precisa che indichi quando normalizzare o standardizzare i dati.

## Gestione delle variabili categoriche

Le variabili/caratteristiche categoriche sono qualsiasi tipo di caratteristica che può essere classificata in due tipi principali:

- Nominali

- Ordinali

Le variabili nominali sono variabili che hanno due o più categorie alle quali non è associato alcun tipo di ordine. Per esempio, se il genere è classificato in due gruppi, cioè maschio e femmina, può essere considerato una variabile nominale. Le variabili ordinali, invece, hanno "livelli" o categorie a cui è associato un ordine particolare. Per esempio, una variabile categoriale ordinale può essere una caratteristica con tre diversi livelli: basso, medio e alto. L'ordine è importante.

Infine abbiamo diviso il dataset in training set (80%) e test set (20%).

### Strumenti utilizzati

Abbiamo sfruttato la libreria sklearn, in particolare le funzioni MinMaxScaler, OneHotEncoder e train_test_split

# Costruzione del modello di tutti i regressori e valutazione delle prestazioni

## Sommario

Abbiamo quindi creato una funzione che richiamerà tutti i modelli di regressione e stamperà a video i punteggi relativi alle loro prestazioni.

Ridge

```
In [98]: predict(Ridge())

Model: Ridge()
Training score: 0.6121844546777211
r2 score: 0.6416289146447629
MAE:22.238793107917086
MSE:961.0544877683288
RMSE:31.000878822516125
```

Lasso

```
In [99]: predict(Lasso())

Model: Lasso()
Training score: 0.520874762902819
r2 score: 0.5200062919536799
MAE:23.997285740370533
MSE:1287.213522712668
RMSE:35.87775805025542
```

Regressione Lineare

```
In [100]: predict(LinearRegression())

Model: LinearRegression()
Training score: 0.6140589142678947
r2 score: 0.6403330418039275
MAE:22.224269070511145
MSE:964.529668831074
RMSE:31.056877963360613
```

Decision Tree Regression

```
In [101]: predict(DecisionTreeRegressor())

          Model: DecisionTreeRegressor()
          Training score: 0.9703744348956336
          r2 score: 0.6658226979444458
          MAE:16.088458472820022
          MSE:896.1732934799936
          RMSE:29.936153618659723
```

Random Forest Regressor

```
In [102]: predict(RandomForestRegressor())

          Model: RandomForestRegressor()
          Training score: 0.9520319149707726
          r2 score: 0.8406890979600953
          MAE:13.051621623023346
          MSE:427.2288240110206
          RMSE:20.669514363211842
```

Gradient Boosting Regressor

```
In [103]: predict(GradientBoostingRegressor())

          Model: GradientBoostingRegressor()
          Training score: 0.7772211257852113
          r2 score: 0.8018998191799738
          MAE:16.693247231679027
          MSE:531.2511962735036
          RMSE:23.048887094033493
```

LGBM Regressor

```
In [104]: predict(LGBMRegressor())

          Model: LGBMRegressor()
          Training score: 0.8528611264132058
          r2 score: 0.8465400225989477
          MAE:13.742018478018695
          MSE:411.5382239276197
          RMSE:20.28640490396511
```

XBG Regressor

```
In [105]: predict(XGBRegressor())
```

```
Model: XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
Training score: 0.9335472338292794
r2 score: 0.8720425327154083
MAE:12.403353153376532
MSE:343.14737768374164
RMSE:18.52423757361532
```

KNeighbors Regressor

```
In [107]: predict(KNeighborsRegressor())
```

```
Model: KNeighborsRegressor()
Training score: 0.8113423898592415
r2 score: 0.7835029299797981
MAE:16.336776661884265
MSE:580.5866857962698
RMSE:24.095366479808305
```

Gaussian NB

```
In [108]: predict(GaussianNB())
```

```
Model: GaussianNB()
Training score: 0.05261269879229941
r2 score: -1.5882064549998618
MAE:48.57340985174557
MSE:6940.87087517934
RMSE:83.31188915862694
```

SVR

```
In [109]: predict(SVR())
```

```
Model: SVR()
Training score: 0.5516149008542333
r2 score: 0.5499549460351889
MAE:20.48393554606296
MSE:1206.8993188501413
RMSE:34.740456514705464
```

Ada Boost Regressor

```
In [110]: predict(AdaBoostRegressor())
```

```
Model: AdaBoostRegressor()
Training score: 0.5370920155507534
r2 score: 0.5668424370863205
MAE:27.24615164519394
MSE:1161.6116276131384
RMSE:34.08242402783491
```

Bagging Regressor

```
In [111]: predict(BaggingRegressor())
```

```
Model: BaggingRegressor()
Training score: 0.9430864464407035
r2 score: 0.8214444028178823
MAE:13.591891325635945
MSE:478.8379001557203
RMSE:21.882365049411828
```

SGD Regressor

```
In [112]: predict(SGDRegressor())
```

```
Model: SGDRegressor()
Training score: 0.5866738688939621
r2 score: 0.6059036818935861
MAE:22.539759328193757
MSE:1056.8599158989296
RMSE:32.50938196734797
```

Cat Boost Regressor

```
In [113]: predict(CatBoostRegressor())
```

```
986:    learn: 15.5625452       total: 2.19s
987:    learn: 15.5598321       total: 2.19s
988:    learn: 15.5545580       total: 2.19s
989:    learn: 15.5478406       total: 2.2s
990:    learn: 15.5444163       total: 2.2s
991:    learn: 15.5409241       total: 2.2s
992:    learn: 15.5383871       total: 2.2s
993:    learn: 15.5357000       total: 2.2s
994:    learn: 15.5305280       total: 2.21s
995:    learn: 15.5235958       total: 2.21s
996:    learn: 15.5211896       total: 2.21s
997:    learn: 15.5180818       total: 2.21s
998:    learn: 15.5131230       total: 2.21s
999:    learn: 15.5079673       total: 2.21s
Training score: 0.90626977850439
r2 score: 0.8743836344891951
MAE:12.847082025750353
MSE:336.8691748432701
RMSE:18.353996154605408
```

Hyperparameter Tuning con CatBoost per migliorare le performance

```
In [114]: from catboost import CatBoostRegressor
catboost = CatBoostRegressor(learning_rate=0.1,random_strength=100)
catboost.fit(X_train, y_train)
predictions = catboost.predict(X_test)
r2score=r2_score(y_test,predictions)
print("r2 score: {}".format(r2score))
```

```
982:    learn: 13.9998637       total: 2s       remaining: 34.7ms
983:    learn: 13.9951132       total: 2.01s    remaining: 32.6ms
984:    learn: 13.9907676       total: 2.01s    remaining: 30.6ms
985:    learn: 13.9878976       total: 2.01s    remaining: 28.5ms
986:    learn: 13.9835224       total: 2.01s    remaining: 26.5ms
987:    learn: 13.9789932       total: 2.01s    remaining: 24.5ms
988:    learn: 13.9741713       total: 2.02s    remaining: 22.4ms
989:    learn: 13.9711571       total: 2.02s    remaining: 20.4ms
990:    learn: 13.9651748       total: 2.02s    remaining: 18.4ms
991:    learn: 13.9607749       total: 2.02s    remaining: 16.3ms
992:    learn: 13.9563133       total: 2.02s    remaining: 14.3ms
993:    learn: 13.9531167       total: 2.03s    remaining: 12.2ms
994:    learn: 13.9481997       total: 2.03s    remaining: 10.2ms
995:    learn: 13.9436767       total: 2.03s    remaining: 8.15ms
996:    learn: 13.9409588       total: 2.03s    remaining: 6.12ms
997:    learn: 13.9374347       total: 2.03s    remaining: 4.08ms
998:    learn: 13.9363554       total: 2.04s    remaining: 2.04ms
999:    learn: 13.9295231       total: 2.04s    remaining: 0us
r2 score: 0.8754321635455239
```

Il punteggio di r2 è leggermente aumentato rispetto al risultato precedente.