Corso di Laurea Magistrale in

Ingegneria Informatica

Online Shoppers Purchasing Intention Dataset

Gruppo composto da:
Marmaglio Simone
Mazzotti Giulia

Ottobre 2021

Indice

1 Introduzione

Il seguente progetto sviluppa nel modo più adeguato possibile la definizione dei modelli di apprendimento relativamente al problema descritto nel *Paragrafo 1.1*, come visto e studiato durante il corso di Machine Learning and Data Mining.

1.1 Descrizione del problema

Il problema trattato nel corrente progetto di Machine Learning e Data Mining si occupa di analizzare diverse informazioni relative al comportamento di clienti che acquistano online.
La raccolta delle informazioni a disposizione è stata effettuata da *Google Analytics*, un servizio web che consente, di analizzare dettagliate statistiche sui visitatori di un sito web. In particolare, conserva le informazioni relative alle sessioni di acquisto/non acquisto in una pagina di e-commerce.
I dati coinvolti hanno permesso di determinare opportuni modelli di apprendimento nell'ambito del marketing, questi hanno il fine ultimo di *predirre* il potenziale acquisto di un cliente in seguito alla visita della pagina di e-commerce considerata.

1.2 Dataset e features

Il dataset si compone complessivamente di 12330 istanze aventi ciascuna 18 features.

Feature	***Tipo***	***Descrizione***
Adnimistrative	Intero	numero di pagine di tipo amministrazione visitate
Administrative_Duration	Float	tempo speso dall'utente in pagine di tipo Administrative
Informational	Intero	numero di pagine di tipo informativo visitate
Informational_Duration	Float	tempo speso dall'utente in pagine di tipo Informational
ProductRelated	Intero	numero di pagine relative al prodotto visitate
ProductRelated_Duration	Float	tempo speso dall'utente in pagine di tipo ProductRelated
BounceRates	Float	percentuale di sessioni in cui la pagina è stata l'unica della sessione
ExitRates	Float	percentuale di sessioni in cui la pagina è stata l'ultima della sessione
PageValues	Float	valore della pagina in base al contributo fornito
SpecialDay	Float	vicinanza del giorno della sessione con un giorno festivo
Month	Object	mese nella quale è effettuata la sessione
OperatingSystems	Intero	sistema operativo utilizzato
Browser	Intero	browser utilizzato
Region	Intero	traffic type
TrafficType	Intero	tipo di traffico registrato
VisitorType	Object	tipo di visitatore, returning, new o other
Weekend	Bool	visita della pagina nel fine settimana
Revenue	Bool	acquisto effettuato

1.3 Obiettivo

L'obiettivo dei modelli definiti è quello di prevedere l'acquisto di un prodotto in una determinata pagina di e-commerce sulla base delle features a disposizione.

2 Pre-processing

2.1 Manipolazione dei dati

Al fine di allenare adeguatamente ogni modello, tutte le features sono state ridefinite nel dominio degli interi. In particolare, le variabili **Month** e **VisitorType**, di tipo *Object* sono state ridefinite mediante il metodo `get_dummies`. Questo ha permesso di generare un DataFrame con nomi di colonna fittizi formati concatenando il nome di colonna originale e ogni valore univoco per la colonna.

2.2 Eliminazione dei dati

Mediante opportune funzioni è stato possibile garantire l'assenza di *missing values* nel dataset a disposizione. Sono comunque state rimosse istanze in quanto: doppie o outliers.

2.2.1 Istanze doppie

Per la rimozione di istanze tra loro uguali e quindi potenzialmente ridondanti per le valutazioni effettuate nei passi successivi, è stata utilzzata la funzione `drop_duplicates`. Sono state rimosse 125 istanze doppie ed è stato mantenuto il primo elemento di ciascuna, in modo tale da evitare la rimozione completa e quindi il rischio di non includere informazioni rilevanti.

2.2.2 Outliers

Si è scelto di rimuovere *evidenti* elementi outliers a partire dall'osservazione delle distribuzioni di dati in relazione alle prime sei features.

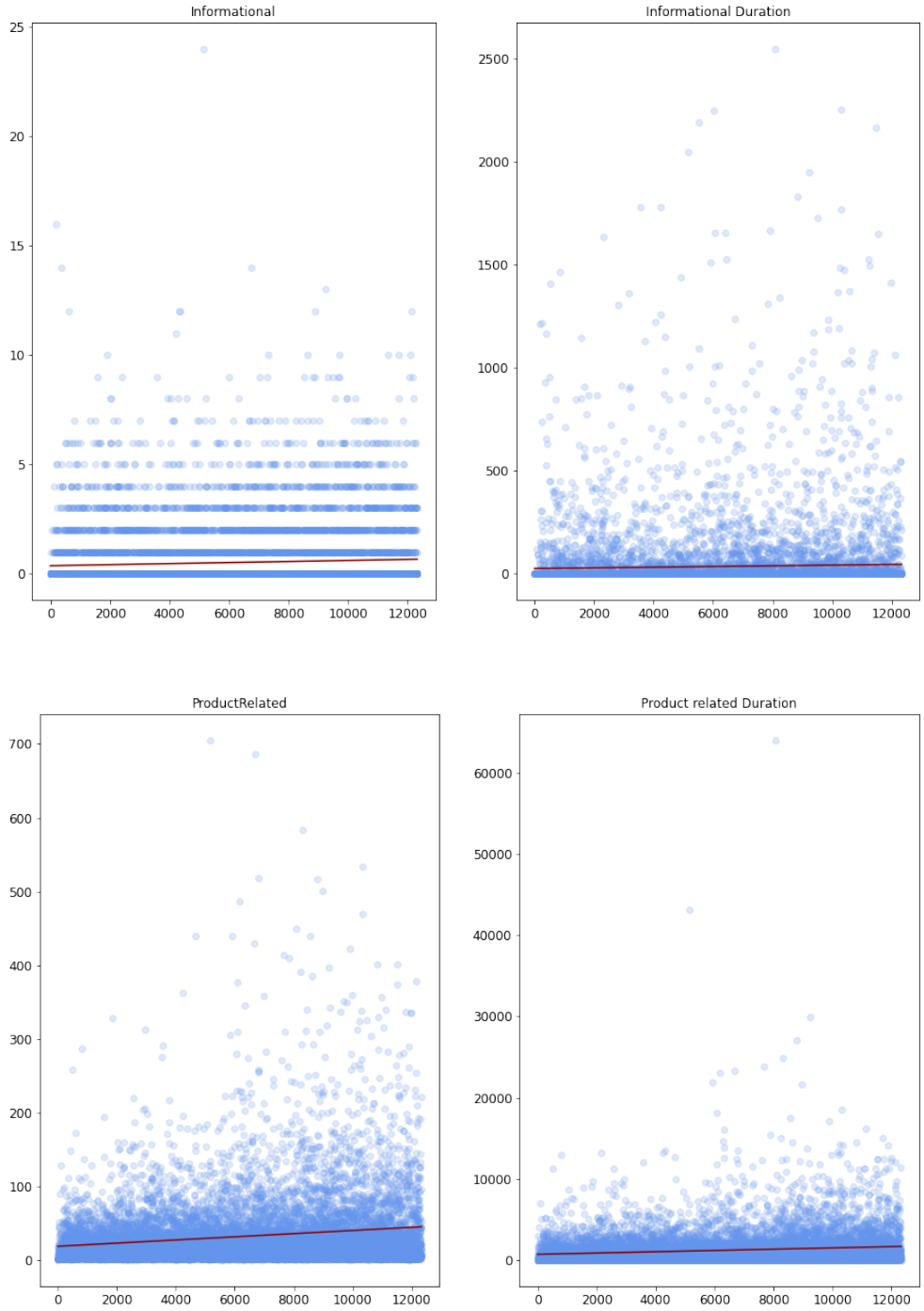


Figura 1: Distribuzione istanze prima della rimozione di outliers

Vista la distribuzione non uniforme delle istanze, un'attenta osservazione ha permesso di individuare i principali valori anomali nella feature *ProductRelated_ Duration*. Pertanto, si è scelto di rimuovere le istanze per cui il valore di tale feature superasse soglia 15000.

3 Analisi dei dati

Si è osservato un importante sbilanciamento del dataset: le istanze classificate negativamente sono risultate molte più rispetto alle istanze classificate positivamente. In particolare, solo il 16% delle sessioni registrate sono terminate con un acquisto effettivo. La problematica di sbilanciamento del dataset è stata opportunamente gestita effettuando nel *Paragrafo 5* operazioni di sampling.

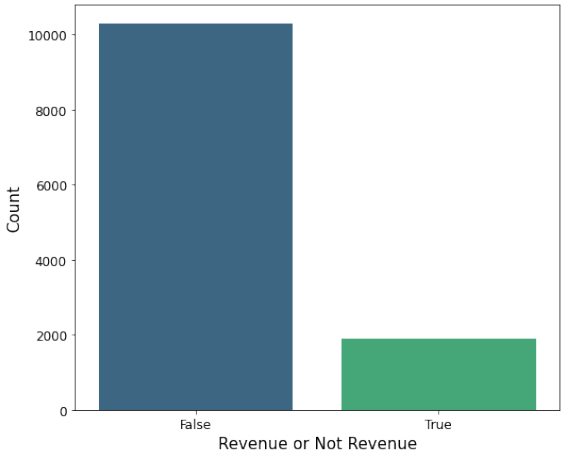


Figura 2: Classificazione delle istanze

In *Figura 4* sono riportate le distribuzioni dei dati in base alle feature indicate.

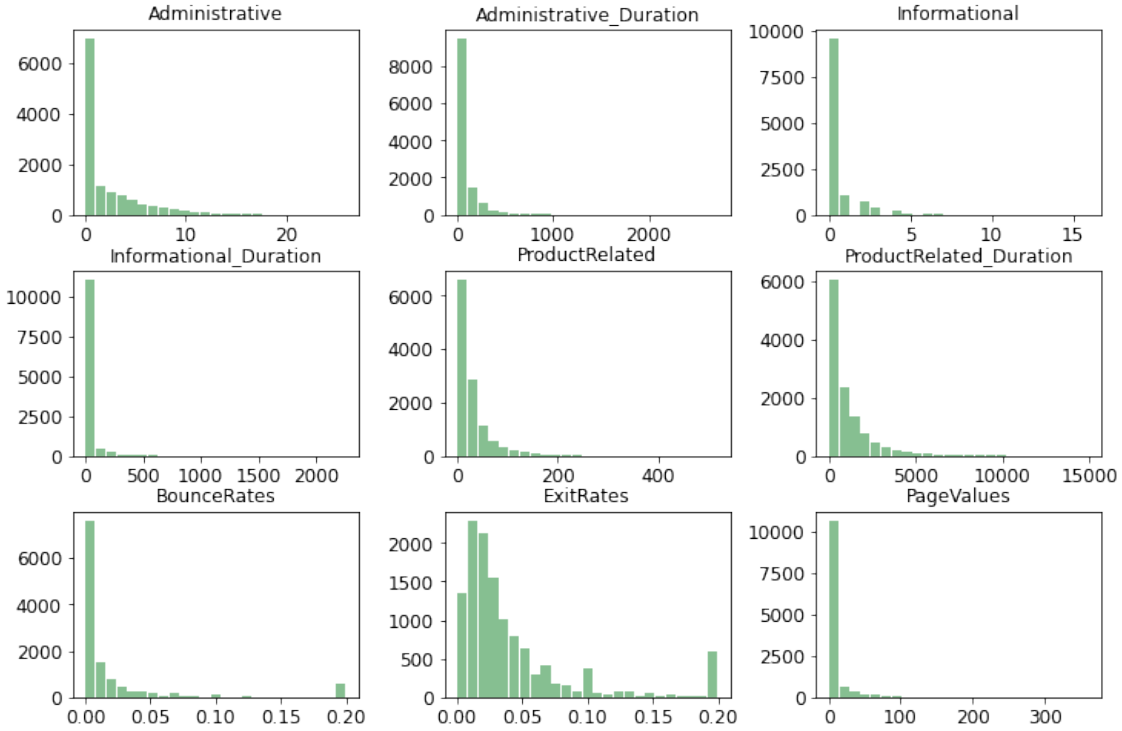


Figura 3: Distribuzione dati

I seguenti grafici permettono di analizzare in modo più approfondito la tipologia delle sessioni registrate. Si nota che una grande maggioranza di utenti si identifica in visitatori *returning*, che avrebbero quindi avviato una sessione di potenziale acquisto in un secondo momento. Inoltre, il grafico relativo ai mesi di acquisto evidenzia che i mesi di Maggio e Novembre sono i preferiti per informarsi relativamente ad alcuni prodotti.

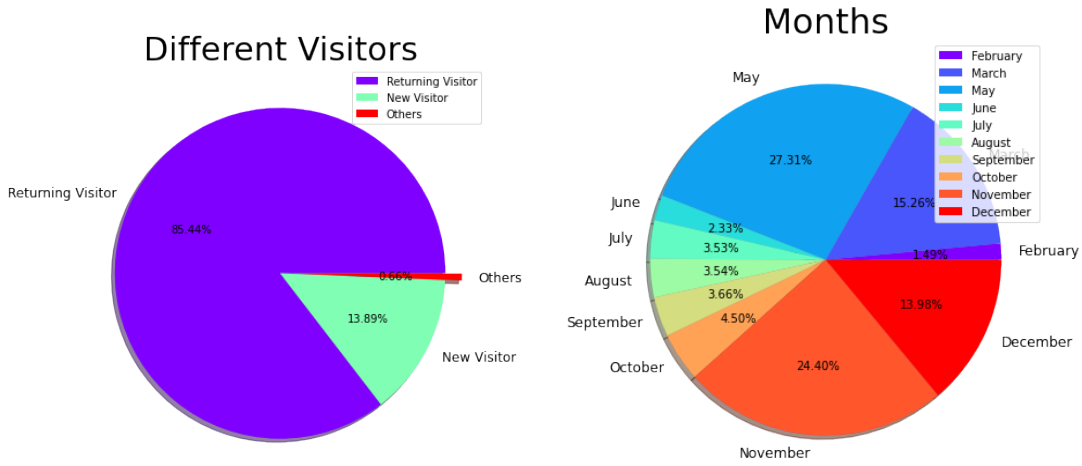


Figura 4: Informazioni aggiuntive

4 Feature Selection

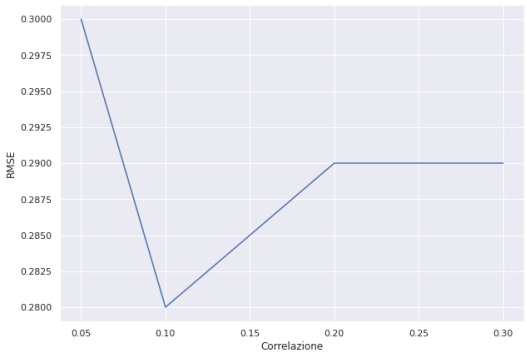
Il processo di **feature selection** ha permesso di ridurre il numero di variabili di input al fine di definire un modello ottimizzandone le prestazioni. Di fatto, riducendo il numero di features si riduce di conseguenza la complessità computazionale e aumentano le performance complessive.
In questa analisi sono state valutate tre diverse tecniche per effettuare questa operazione.

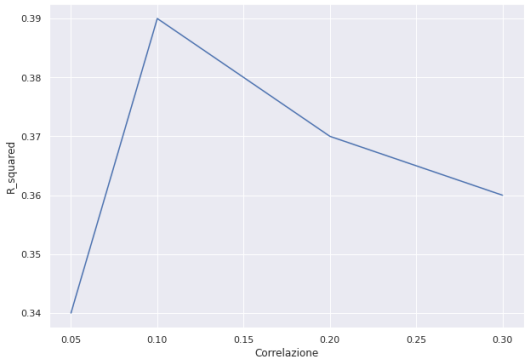
4.1 Correlazione

Il **coefficiente di correlazione** è un valore compreso tra -1 e 1 e viene utilizzato per quantificare il modo nella quale cambiano due variabili, l'una rispetto all'altra. I valori positivi indicano l'esistenza di una *correlazione lineare positiva*, i valori negativi indicano una *correlazione negativa*, il valore 0 indica *assenza di correlazione*.

Sulla base dei valori di correlazione individuati e con un'adeguato utilizzo delle metriche RMSE ed R-squared, sono state selezionate con questo metodo le features: 'Administrative', 'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues', 'Month_Nov', 'VisitorType_New_Visitor', 'VisitorType_Returning_Visitor', in quanto:

- Presentano *tutte* un valore di correlazione, rispetto alla variabile target 'Revenue', maggiore di 0.1.
- Utilizzando le metriche sopra indicate, individuano i valori migliori:
  - RMSE (= indice di distanza tra i valori predetti e quelli osservati, da minimizzare): 0.29
  - R_squared (= percentuale della varianza della variabile indipendente rispetto alla variabile indipendente, da massimizzare): 0.38





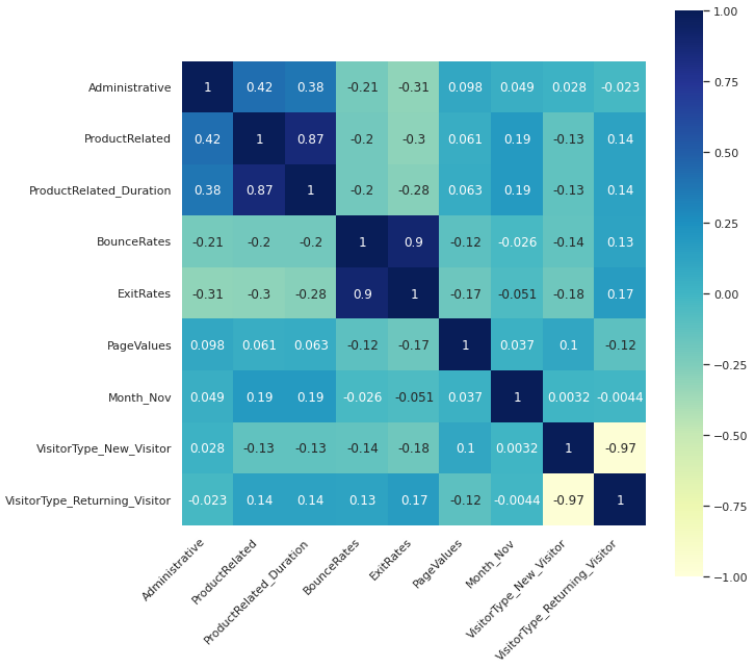


Figura 5: Correlazione tra le features

4.2 Select K Best

Data una *score function*, la tecnica **Select K Best** permette di selezionare le prime K features con score più alto.
Le funzioni utilizzate per questo progetto sono:

- χ^2 = calcola la statistica χ^2 per ciascuna feature di X e y. Ottenendo uno score piccolo, significa che y è indipendente dalla feature in esame, un valore alto indica invece che esiste una certa relazione tra la feature ed il valore target.
- $F - statistic$ = sfrutta la regressione lineare per confrontare le varianze di ciascuna feature rispettivamente alla variabile target. Quindi, utilizza l'esito come criterio per individuare le features utili al compito di classificazione delle istanze rispetto al target.

Utilizzando le due score function appena descritte e scegliendo come valore K il valore 5, si ottengono rispettivamente:

- 'Administrative_Duration', 'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration', 'PageValues'
- 'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues'

4.3 Recursive Feature Elimination

RFE è un algoritmo di selezione delle feature molto noto, in quanto facile da configurare e dall'esito efficace: permette di selezionare le features più rilevati nella predizione del valore target. Per effettuare questa operazione è necessario definire il modello con la quale si esegue l'analisi di predizione, nel progetto corrente è stato scelto un classificatore *Random Forest*. L'esito ottenuto è il seguente

RANKING OF FEATURES [1 1 1 1 1 1 1 1 1 10 1 1 1 1 3 12 6 15 11 13 7 4 1 8 9 5 14 2]

TOP RANKED FEATURES 'Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration', 'BounceRates', 'ExitRates', 'PageValues', 'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'Month_Nov'

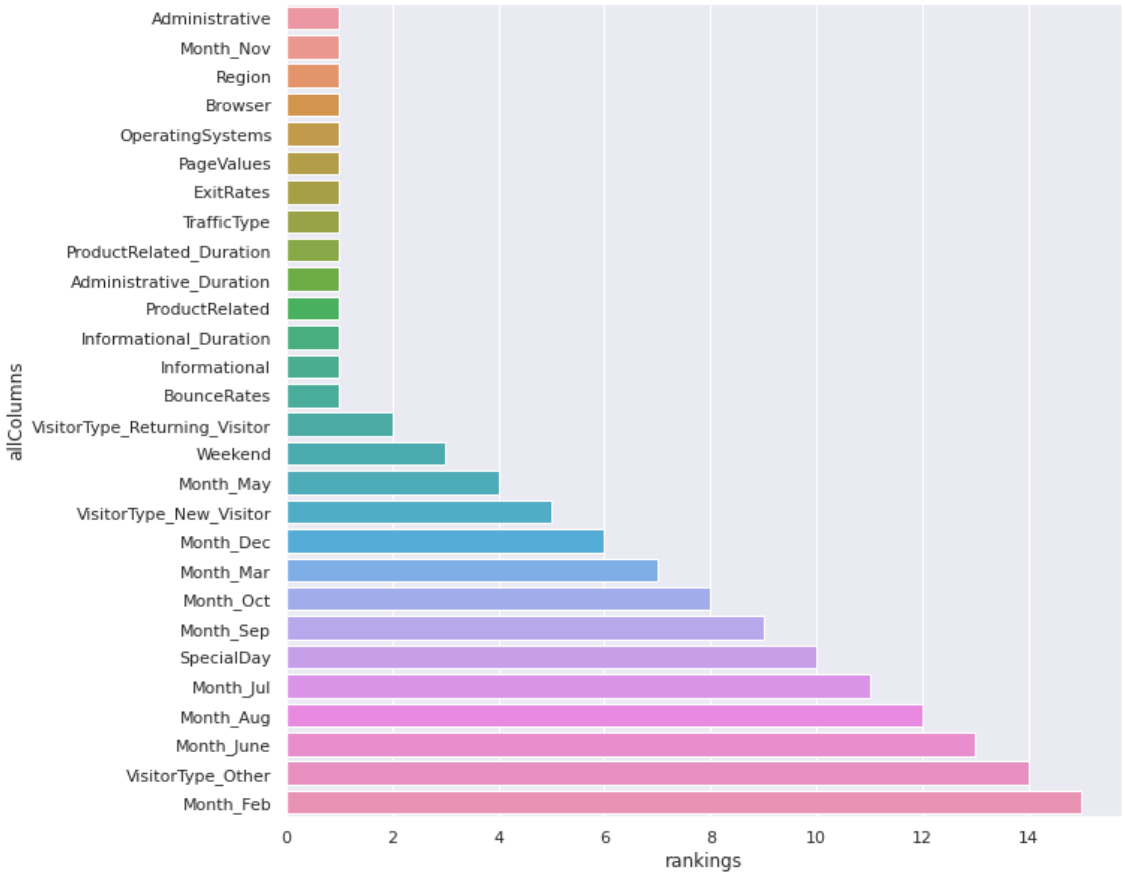


Figura 6: Ranking features

4.4 Scelta

Si è scelto di mantenere la selezione di features ottenuta sfruttando il coefficiente di correlazione, in quanto sia esaustivo dal punto di vista di significato che completo considerando la relazione con la variabile target.

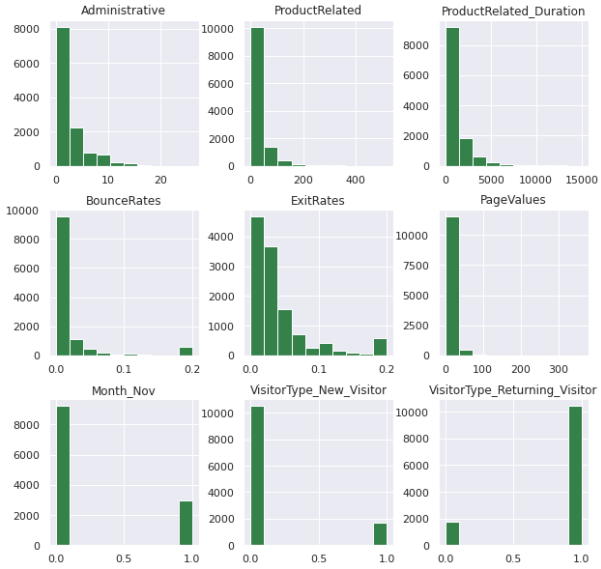


Figura 7: Distribuzione delle istanze per le features selezionate

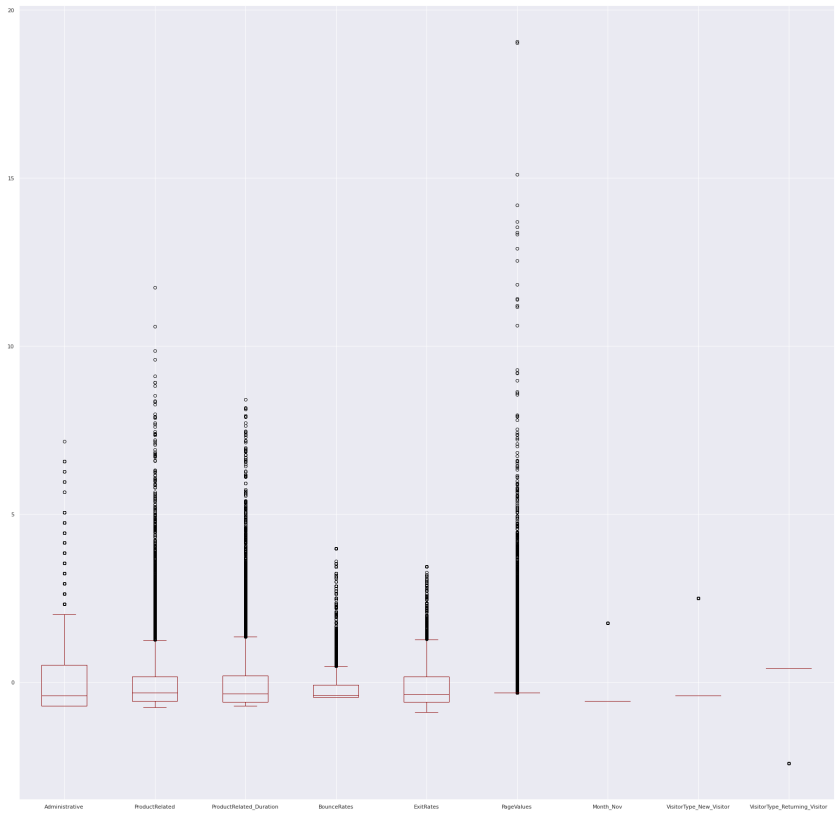


Figura 8: Distribuzione delle istanze per le features selezionate

5 Gestione dei dati

Visto lo sbilanciamento del dataset per quanto riguarda le due classi target, si è deciso di procedere con operazioni di sampling al fine di ottenere un insieme di istanze più completo e coerente. Dal momento che l'obiettivo è quello di prevedere il potenziale acquisto durante una sessione è necessario aumentare le istanze classificate in modo positivo, per minimizzare il numero di falsi positivi risultanti dalle previsioni. Si sono adottate due diverse tecniche.

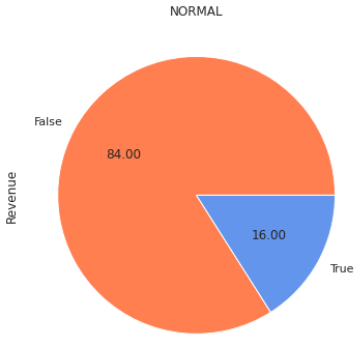


Figura 9: Percentuale di sbilanciamento del dataset

5.1 SMOTE Oversampling

La prima tecnica utilizzata per il bilanciamento delle classi è *SMOTE*, ovvero Synthetic Minority Oversampling Technique, ed è tra le più comunemente utilizzate allo scopo. In particolare, risolve il problema *sovracampionando* gli esempi nella classe di minoranza, in modo casuale. La sua implementanzione prevede l'uso di un algoritmo **K-Nearest Neighbors**: sceglie dati casuali della classe minoritaria (in questo caso, `True`), imposta i K vicini più prossimi da tali osservazioni e definisce questi come *dati sintetici*. Tali dati sono creati tra dati casuali e i vicini dell'istanza selezionata, in modo casuale.
In particolare, si è usata la tecnica *SMOTENC*, che prende in considerazione l'esistenza di variabili categoriche.

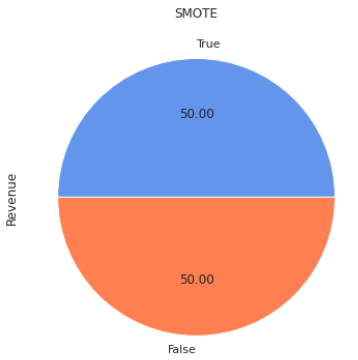


Figura 10: Bilanciamento delle istanze mediante il metodo SMOTENC

5.2 Sampling combinato

È interessante notare che metodi di Sampling delle istanze del dataset possono essere combinati, in questo progetto viene utilizzata la tecnica combinazione del metodo *SMOTE* (*Paragrafo 5.1*) con il metodo di *Undersampling randomico*. Questo permette di bilanciare il dataset a disposizione eliminando *randomicamente* tante istanze della classe maggioritaria fino ad arrivare ad un numero uguale alle istanze della classe minoritaria.
Si tratta di un metodo definibile come *naive* in quanto non prevede nessuna assunzione relativamente alla forma dei dati e non utilizza euristiche di alcun tipo. In questo modo, l'implementazione è più facile e veloce da eseguire, preferibile in casi con un grande numero di istanze.

La combinazione delle due tecniche descritte prevede l'applicazione del metodo *SMOTE* sulla classe minoritaria e quindi un *Undersampling* sulla classe maggioritaria.

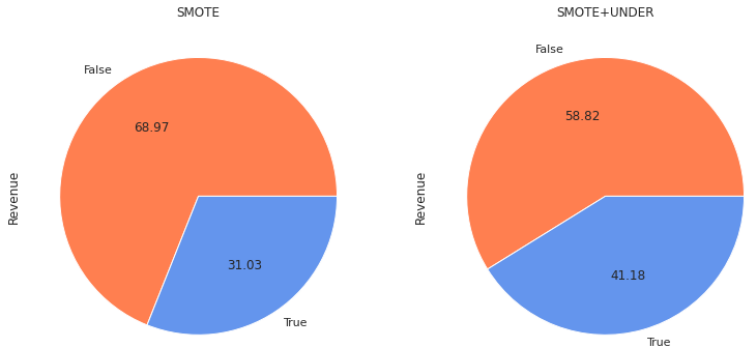


Figura 11: Bilanciamento del dataset mediante sampling combinato

6 Modelli

6.1 Hyperparameter tuning

Per l'operazione di sincronizzazione degli iperparametri si è scelto l'algoritmo *grid search*, implementato come segue.

```
def search(model, param_grid, X_train, y_train,label):
  print('\n\nTRAINING MODEL', label)
  cv_method = StratifiedKFold(n_splits=3)

  clf = GridSearchCV(
        estimator=model,
        param_grid=param_grid,
        scoring='precision',
        n_jobs=-1,
        cv=cv_method,
        verbose=0,
        return_train_score=True,
  )

  clf.fit(X_train, y_train)

  return clf
```

6.2 Valutazione dei modelli

Ciascuna classe di modelli scelta è stata allenata sui tre dataset ottenuti mediante le operazioni di gestione dei dati, viste nel *Paragrafo 5*: dataset *NORMAL*, senza alcun bilanciamento, *SMOTE*, dataset bilanciato con il metodo di Oversampling, e *SMOTE + UNDER*, definito con la combinazione dei due metodi di sampling.

Le seguenti porzioni di codice riportano le funzioni impiegate per l'addestramento dei modelli e la relativa valutazione.

```
def stat_test(model, X_test, y_test,label):
    print('\n\nRISULTATI MODELLO CON DATASET', label)
    #Testing the model
    cfmatrix=confusion_matrix(y_true=y_test, y_pred=model.predict(X_test))
    print('Confusion Matrix: ', cfmatrix)
    #Defining prints for accuracy metrics of grid
    print("\n**Grid search results of","**")
    print("The best parameters are:",model.best_params_)
    print("Best training accuracy:\t", model.best_score_)
    print('Classification Report:')
    print(classification_report(y_true=y_test, y_pred=model.predict(X_test)))
```

```
def confronto_mod(vet_X_train,vet_y_train,X_test,y_test, model, param_grid, label):
  clf = []
  for i in range(len(vet_X_train)):
    clf.append(search(model, param_grid, vet_X_train[i],vet_y_train[i],label[i]))

  for i in range(len(clf)):
    stat_test(clf[i],X_test,y_test,label[i])

  return clf
```

6.3 Addestramento dei modelli

Di seguito, per ogni modello definito, sono riportati i risultati ottenuti, si fa particolare riferimento ai valori dei parametri: **precision** e **recall**, ottenuti mediante la definizione della relativa matrice di confusione.

6.3.1 Support Vector Machines

Risultati modello con dataset *normal*				
Confusion matrix	3571	58		
	453	184		
Best parameters	'C': 0.1, 'gamma': 1, 'kernel': 'rbf', 'random_state': 42			
Best training accuracy	0.80			
Classification report				
	Precision	Recall	F1-score	Support
False	0.89	0.98	0.93	3629
True	0.76	0.29	0.42	637
Accuracy			0.88	4266
Macro avg	0.82	0.64	0.68	4266
Weighted avg	0.87	0.88	0.86	4266

Risultati modello con dataset *smote*				
Confusion matrix	3270	359		
	166	471		
Best parameters	'C': 0.1, 'gamma': 0.1, 'kernel': 'linear', 'random_state': 42			
Best training accuracy	0.90			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.90	0.93	3629
True	0.57	0.74	0.64	637
Accuracy			0.88	4266
Macro avg	0.76	0.82	0.78	4266
Weighted avg	0.89	0.88	0.88	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3463	166		
	382	255		
Best parameters	'C': 0.1, 'gamma': 10, 'kernel': 'rbf', 'random_state': 42			
Best training accuracy	0.96			
Classification report				
	Precision	Recall	F1-score	Support
False	0.90	0.95	0.93	3629
True	0.61	0.40	0.48	637
Accuracy			0.87	4266
Macro avg	0.75	0.68	0.70	4266
Weighted avg	0.86	0.87	0.86	4266

6.3.2 Gaussian Naive Bayes

Risultati modello con dataset *normal*				
Confusion matrix	3512	117		
	462	175		
Best parameters	'var_smoothing': 1.0			
Best training accuracy	0.636			
Classification report				
	Precision	Recall	F1-score	Support
False	0.88	0.97	0.92	3629
True	0.60	0.27	0.38	637
Accuracy			0.83	4266
Macro avg	0.74	0.62	0.65	4266
Weighted avg	0.84	0.86	0.84	4266

Risultati modello con dataset *smote*				
Confusion matrix	3172	457		
	259	378		
Best parameters	'var_smoothing': 1.0			
Best training accuracy	0.83			
Classification report				
	Precision	Recall	F1-score	Support
False	0.92	0.87	0.90	3629
True	0.45	0.59	0.51	637
Accuracy			0.83	4266
Macro avg	0.69	0.73	0.71	4266
Weighted avg	0.85	0.83	0.84	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3412	217		
	365	272		
Best parameters	'var_smoothing': 1.0			
Best training accuracy	0.84			
Classification report				
	Precision	Recall	F1-score	Support
False	0.90	0.94	0.92	3629
True	0.56	0.43	0.48	637
Accuracy			0.86	4266
Macro avg	0.73	0.68	0.70	4266
Weighted avg	0.85	0.86	0.86	4266

6.3.3 Adaboost

Risultati modello con dataset *normal*				
Confusion matrix	3480	149		
	319	318		
Best parameters	'algorithm': 'SAMME.R', 'learning_rate': 0.01, 'n_estimators': 500, 'random_state': 42			
Best training accuracy	0.73			
Classification report				
	Precision	Recall	F1-score	Support
False	0.92	0.96	0.94	3629
True	0.68	0.50	0.58	637
Accuracy			0.89	4266
Macro avg	0.80	0.73	0.76	4266
Weighted avg	0.88	0.89	0.88	4266

Risultati modello con dataset *smote*				
Confusion matrix	3290	339		
	166	471		
Best parameters	'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 1000, 'random_state': 42			
Best training accuracy	0.92			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.91	0.93	3629
True	0.58	0.74	0.65	637
Accuracy			0.88	4266
Macro avg	0.77	0.82	0.79	4266
Weighted avg	0.90	0.88	0.89	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3273	356		
	165	472		
Best parameters	'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 1000, 'random_state': 42			
Best training accuracy	0.87			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.90	0.93	3629
True	0.57	0.74	0.64	637
Accuracy			0.88	4266
Macro avg	0.76	0.82	0.79	4266
Weighted avg	0.89	0.88	0.88	4266

6.3.4 Extreme Gradient Boosting

Risultati modello con dataset *normal*				
Confusion matrix	3349	280		
	292	345		
Best parameters	'algorithm': 'SAMME', 'base_estimator': DecisionTreeClassifier(), 'learning_rate': 0.001, 'n_estimators': 1000, 'random_state': 42			
Best training accuracy	0.58			
Classification report				
	Precision	Recall	F1-score	Support
False	0.92	0.92	0.92	3629
True	0.55	0.54	0.55	637
Accuracy			0.87	4266
Macro avg	0.74	0.73	0.73	4266
Weighted avg	0.86	0.87	0.87	4266

Risultati modello con dataset *smote*				
Confusion matrix	3211	418		
	239	398		
Best parameters	'algorithm': 'SAMME', 'base_estimator': DecisionTreeClassifier(), 'learning_rate': 0.01, 'n_estimators': 1000, 'random_state': 42			
Best training accuracy	0.88			
Classification report				
	Precision	Recall	F1-score	Support
False	0.93	0.88	0.91	3629
True	0.49	0.62	0.55	637
Accuracy			0.85	4266
Macro avg	0.71	0.75	0.73	4266
Weighted avg	0.86	0.85	0.85	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3188	441		
	208	429		
Best parameters	'algorithm': 'SAMME', 'base_estimator': DecisionTreeClassifier(), 'learning_rate': 0.1, 'n_estimators': 50, 'random_state': 42			
Best training accuracy	0.81			
Classification report				
	Precision	Recall	F1-score	Support
False	0.94	0.88	0.91	3629
True	0.49	0.67	0.57	637
Accuracy			0.85	4266
Macro avg	0.72	0.78	0.74	4266
Weighted avg	0.87	0.85	0.86	4266

6.3.5 Random Forest Classifier

Risultati modello con dataset *normal*				
Confusion matrix	3505	124		
	299	338		
Best parameters	'criterion': 'gini', 'max_depth': 5,'n_estimators': 100, 'n_jobs' = -1, 'random_state': 42			
Best training accuracy	0.78			
Classification report				
	Precision	Recall	F1-score	Support
False	0.92	0.97	0.94	3629
True	0.73	0.53	0.62	637
Accuracy			0.90	4266
Macro avg	0.83	0.75	0.78	4266
Weighted avg	0.89	0.90	0.89	4266

Risultati modello con dataset *smote*				
Confusion matrix	3276	354		
	185	452		
Best parameters	'criterion': 'gini', 'max_depth': 25,'n_estimators': 100, 'n_jobs' = -1, 'random_state': 42			
Best training accuracy	0.91			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.90	0.92	3629
True	0.56	0.71	0.63	637
Accuracy			0.87	4266
Macro avg	0.75	0.81	0.78	4266
Weighted avg	0.89	0.87	0.88	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3272	357		
	159	478		
Best parameters	'criterion': 'entropy', 'max_depth': 25,'n_estimators': 100, 'n_jobs' = -1, 'random_state': 42			
Best training accuracy	0.87			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.90	0.93	3629
True	0.57	0.75	0.65	637
Accuracy			0.88	4266
Macro avg	0.76	0.83	0.79	4266
Weighted avg	0.90	0.88	0.89	4266

6.3.6 Neural Nets

Risultati modello con dataset *normal*				
Confusion matrix	3476	153		
	285	352		
Best parameters	'alpha': 0.01, 'hidden_layer_sizes': 150, 'learning_rate': 'invscaling', 'max_iter' = 300, 'momentum': 0.9, 'validation_fraction': 0.1			
Best training accuracy	0.74			
Classification report				
	Precision	Recall	F1-score	Support
False	0.92	0.96	0.94	3629
True	0.70	0.55	0.62	637
Accuracy			0.90	4266
Macro avg	0.81	0.76	0.78	4266
Weighted avg	0.89	0.89	0.89	4266

Risultati modello con dataset *smote*				
Confusion matrix	3225	404		
	148	352		
Best parameters	'alpha': 0.01, 'hidden_layer_sizes': 150, 'learning_rate': 'invscaling', 'max_iter' = 300, 'momentum': 0.9, 'validation_fraction': 0.1			
Best training accuracy	0.89			
Classification report				
	Precision	Recall	F1-score	Support
False	0.96	0.89	0.92	3629
True	0.55	0.77	0.64	637
Accuracy			0.87	4266
Macro avg	0.75	0.83	0.78	4266
Weighted avg	0.90	0.87	0.88	4266

Risultati modello con dataset *smote + under*				
Confusion matrix	3253	376		
	159	478		
Best parameters	'alpha': 0.01, 'hidden_layer_sizes': 150, 'learning_rate': 'invscaling', 'max_iter' = 200, 'momentum': 0.9, 'validation_fraction': 0.1			
Best training accuracy	0.86			
Classification report				
	Precision	Recall	F1-score	Support
False	0.95	0.90	0.92	3629
True	0.56	0.75	0.64	637
Accuracy			0.87	4266
Macro avg	0.76	0.82	0.78	4266
Weighted avg	0.89	0.87	0.88	4266

6.4 Conclusioni

Il valore di *accuracy*, evidenziato per ognuno dei modelli allenati, rappresenta la valutazione delle performance presa in considerazione. In generale, non si è ottenuto nessun modello più performante rispetto agli altri. Il valore massimo è stato ottenuto con i metodi ensemble e con le reti neurali, che trovano i loro punti di forza nella robustezza e nelle buone prestazioni ottenute complessivamente.

Inoltre, quasi tutti i modelli individuati, vedono il valore di *accuracy* della fase di training inferiore rispetto a quello della fase di testing; si può concludere che non si hanno situazioni di *overfitting* rispetto ai dati con la quale vengono addestrati i modelli.

L'aspetto non positivo del dataset utilizzato, lo sbilanciamento dei dati, si nota ampiamente nella matrice di confusione riportata per ciascuno dei modelli. Mentre si ha una buona classificazione delle istanze negative, circa gli utenti che non hanno effettuato un acquisto, i modelli ottenuti tendono a *sovrastimare* la presenza di istanze positive. Queste indicano gli utenti che hanno effettuato un acquisto nella sessione registrata: ottenere Falsi Positivi è quindi potenzialmente svantaggioso. Queste osservazioni sono confermate dai valori di precision e recall, "riassunti" quindi dal valore F1-Score, che rappresenta la media armonica delle prime due. Esso tende ad essere maggiore per le istanze negative, ad indicare una buona classificazione di queste, e molto minore per le istanze positive, a significare che di tutte le istanze classificate positivamente dal modello, una buona percentuale non è predetta in modo adeguato.

In particolare, l'addestramento senza l'implementazione di tecniche di balancing dei dati porta a modelli con *accuracy* inferiore sul training set, mentre è maggiore sul test set. Oltre a ciò, si può notare che per la classificazione delle istanze positive la *precision* è maggiore rispetto ai modelli ottenuti con il bilanciamento dei dati. Tuttavia, per questi ultimi aumentano i valori di *recall* ed *F1-Score*.

Sono riportati nelle seguenti tabelle i risultati complessivi analizzati per la valutazione delle performance di ciascun modello.

NORMAL						
	SVM	**GNB**	**AB**	**EGB**	**RFC**	**NN**
Training accuracy	80%	64%	73%	58%	78%	74%
Test accuracy	88%	83%	89%	87%	90%	90%
Precision	76%	60%	68%	55%	73%	70%
Recall	29%	27%	50%	54%	53%	55%
F1-Score	42%	38%	58%	55%	62%	62%

SMOTE						
	SVM	**GNB**	**AB**	**EGB**	**RFC**	**NN**
Training accuracy	90%	83%	92%	88%	91%	89%
Test accuracy	88%	83%	88%	85%	87%	87%
Precision	57%	45%	58%	49%	56%	55%
Recall	74%	59%	74%	62%	71%	77%
F1-Score	64%	51%	65%	55%	63%	64%

SMOTE e UNDERSAMPLING						
	SVM	**GNB**	**AB**	**EGB**	**RFC**	**NN**
Training accuracy	96%	84%	87%	81%	87%	86%
Test accuracy	87%	86%	88%	85%	88%	87%
Precision	61%	56%	57%	49%	57%	56%
Recall	40%	43%	74%	67%	75%	75%
F1-Score	48%	48%	64%	57%	65%	64%

7 Possibili miglioramenti: feature expansion, selection e Optuna

Per un'ulteriore analisi, sono state generate, sulla base di quelle esistenti, nuove features al fine di addestrare nuovamente i modelli. Per l'addestramento è stato usato il framework Optuna al fine di ottenere una rete neurale ottimizzata.

7.1 Calcolo della media

In primo luogo, è stato calcolato il valore medio delle features relative alle tipologie di pagine visitate rispetto al tempo speso su esse.

- $\frac{\texttt{Administrative\_Duration}}{\texttt{Administrative}} = \texttt{mean\_Adm}$; tempo medio dell'utente speso su una pagina di tipo Administrative
- $\frac{\texttt{Informational\_Duration}}{\texttt{Informational}} = \texttt{mean\_Info}$; tempo medio dell'utente speso su una pagina di tipo Informational
- $\frac{\texttt{ProductRelated\_Duration}}{\texttt{ProductRelated}} = \texttt{mean\_PR}$; tempo medio dell'utente speso su una pagina di tipo Product Related

Date le feature ottenute, che, per ogni istanza valutano il valore medio, si possono potenzialmente ottenere gruppi di utenti coerenti rispetto al tempo speso sulle diverse tipologie pagine.

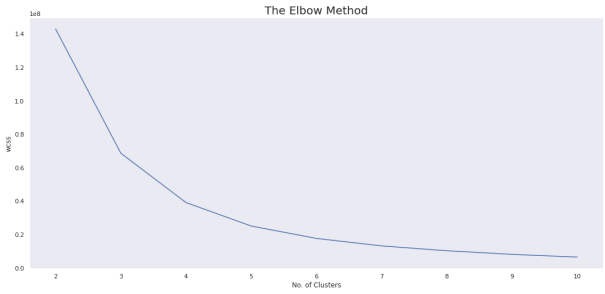
7.2 Duration vs Bounce Rates

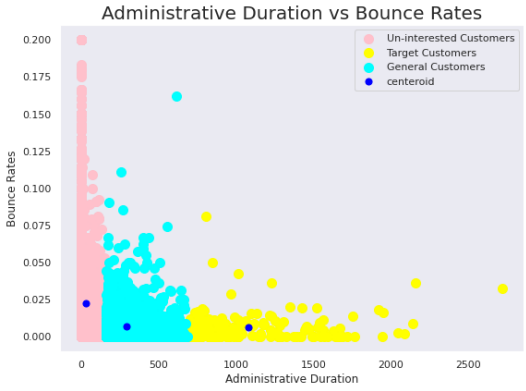
Confrontando le features di Duration relative a ciascuna tipologia di pagina con il valore Bounce è stato implementato il metodo KMeans. In particolare, sono state individuate tre nuove features: `Un-interested customers`, `Target customers` e `General customers`. Vengono riportati i parametri utilizzati per la ricerca del numero ottimale di clusters. Le tre diverse categorie sono state definite in questo modo in quanto si ipotizza che a parità di tempo speso sulla pagina, sessioni con Bounce Rate superiore, non tendano a terminare con un acquisto.
Il valore di `clusters` varia sulla base del valore individuato mediante l'*elbow method*.

```
km = KMeans(n_clusters = clusters,
            init = 'k-means++',
            max_iter = 300,
            n_init = 10,
            random_state = 0,
            algorithm = 'elkan',
            tol = 0.001)
```

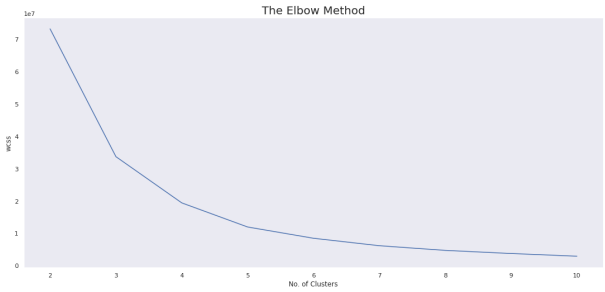
La classe di appartenenza risultante dell'istanza è stata, per ciascuna delle tre features, mantenuta in una nuova feature, rispettivamente introdotta come `Cust_Admin`, `Cust_Info` e `Cust_PR`. Infine è stata definita la nuova feature come valore medio delle tre appena introdotte.

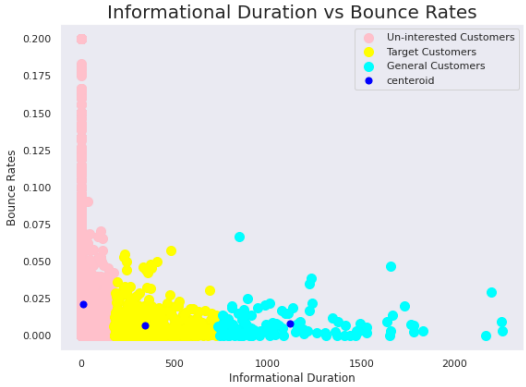
7.2.1 Administrative Duration



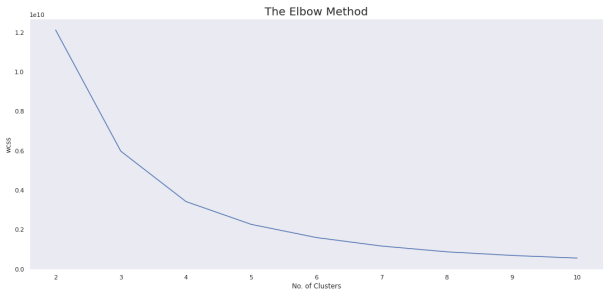


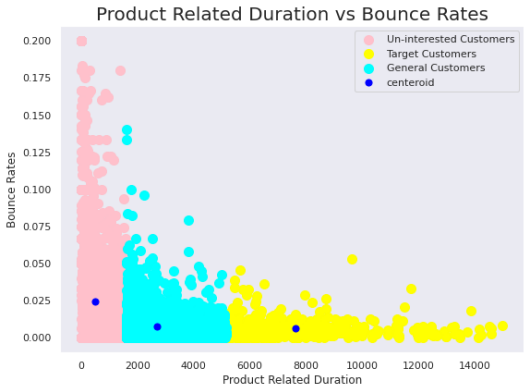
7.2.2 Informational Duration





7.2.3 Product Related Duration





7.3 Feature selection

Unendo al dataset originario le feature introdotte nei paragrafi precedenti, si è implementato il seguente pseudocodice, mediante la quale sono state selezionate le feature con la quale sono stati ottenuti i valori ottimali per le metriche RMSE ed R_squared.

```
F_min_RMSE =[]
F_max_R = []
```

```
min_RMSE = 1
max_R = 0
for j in range(1,X.shape[1]):

  for primo in range(0,X.shape[1]-1):
      if j+primo<=X.shape[1]:
    #if j!=1 and primo!=1:
        lista = []
        for i in range(primo,j+primo-1):

          lista.append(X.columns.values[i])

        for i in range(primo+j-1,X.shape[1]):
          lista_cop = lista_copia(lista)
          lista_cop.append(X.columns.values[i])
          X_f=X[lista_cop]

          y_pred = cross_val_predict(classifier_pipeline, X_f, y, cv=cv)
          RMSE = round(sqrt(mean_squared_error(y,y_pred)),5)
          R_sq = round(r2_score(y,y_pred),5)
          if RMSE<min_RMSE:
            min_RMSE = RMSE
            F_min_RMSE = lista_cop

          if R_sq>max_R:
            max_R = R_sq
            F_max_R = lista_cop
```

Si sono ottenuti valori ottimali per: ['ProductRelated_Duration', 'BouceRates', 'ExitRates', 'PageValues', 'Month_Nov']

7.4 Risultati ottenuti

Dopo aver effettuato la forward feature selection descritta e dopo aver verificato le migliori prestazioni dalla combinazione di ciò che è stato ottenuto, si è notato che non si ottengono feature diverse da quelle già considerate per i diversi modelli di apprendimento.

7.5 Optuna

Le features selezionate con il metodo descritto al *Paragrafo 8.3* sono state utilizzate per la restrizione del dataset originario con la quale è stato implementato il framework Optuna. Sono stati allenati diversi modelli e tra essi è stato estratto il migliore in termini di ottimizzazione di una funzione obiettivo. Di seguito sono riportati i risultati ottenuti per il dataset a disposizione.

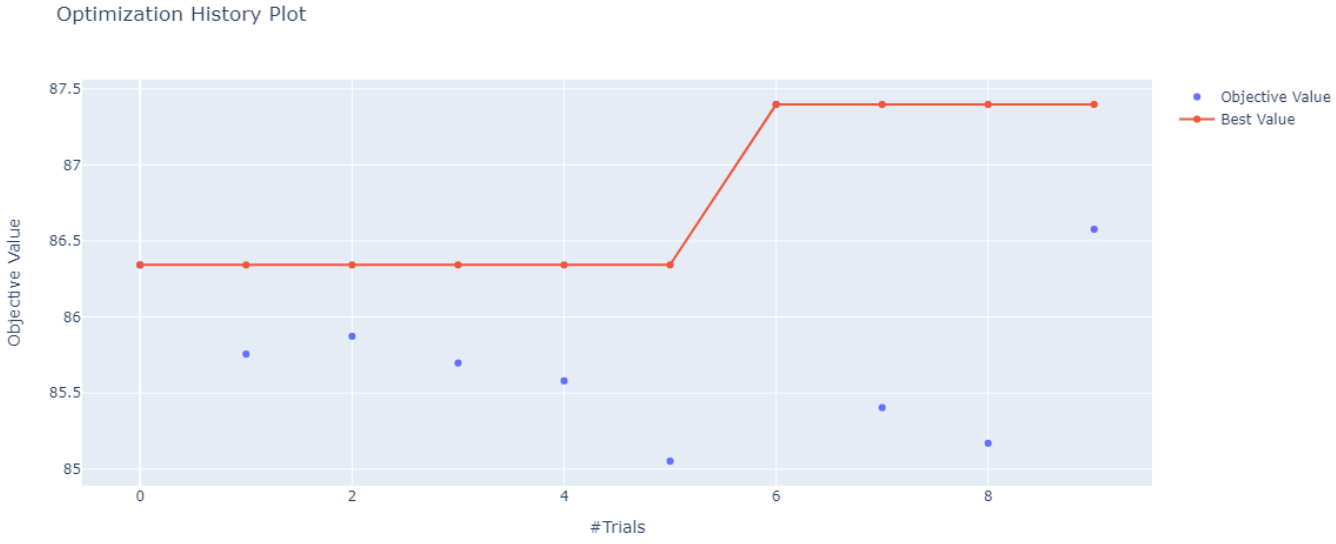
Confusion matrix (training set)		
	F	T
F	5023	685
T	187	879

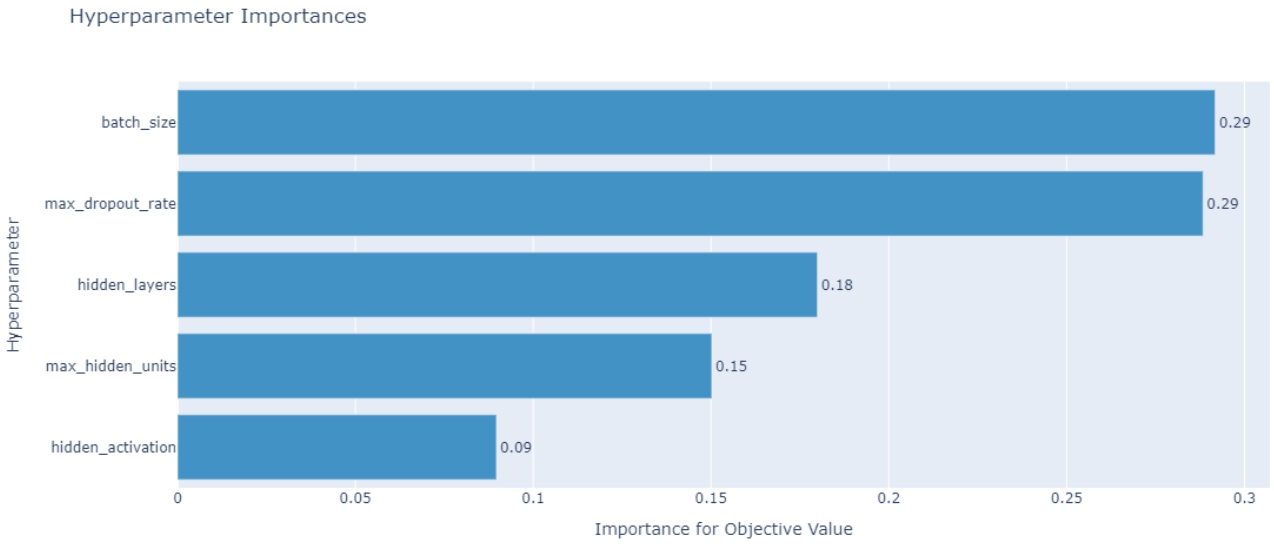
Accuracy	87%
Precision	56%
Recall	82%
F1-Score	66%

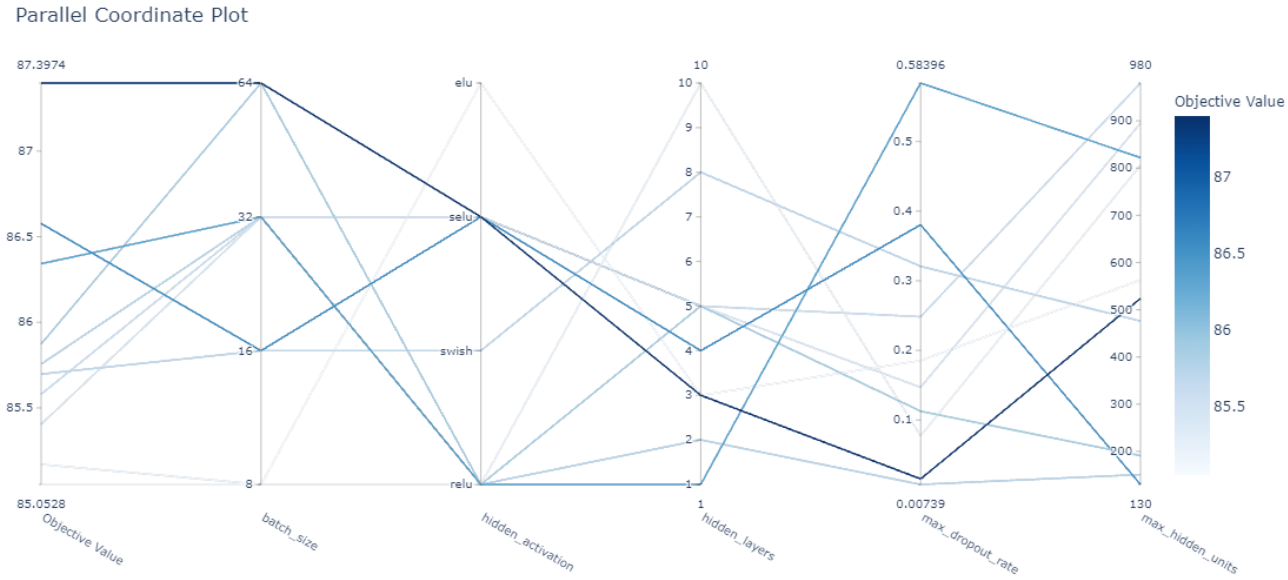
Confusion matrix (training set)		
	F	T
F	5023	685
T	187	879

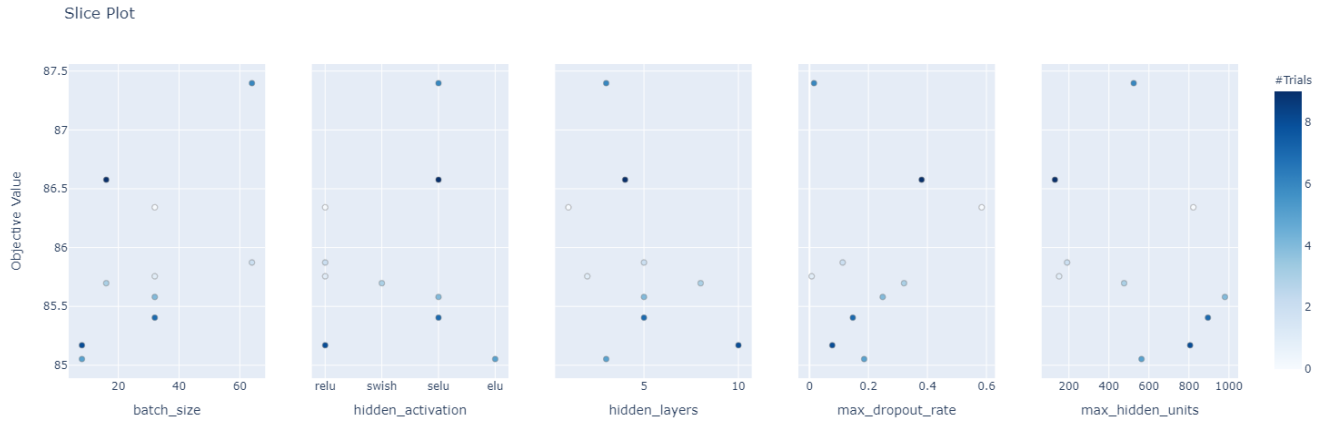
Accuracy	87%
Precision	56%
Recall	82%
F1-Score	66%

number	value	activity regularizer	batch norm	batch size	hidden activation	hidden layers	max dropout rate	max hidden units
6	87.39	True	True	64	selu	3	0.015	524
9	86.57	False	True	16	selu	4	0.38	130
0	86.34	True	False	32	relu	1	0.58	822
2	85.87	False	True	64	relu	5	0.11	191
1	85.75	False	False	32	relu	2	0.007	151
3	85.69	False	True	16	swish	8	0.32	476
4	85.58	False	True	32	selu	5	0.25	980
7	85.40	True	False	32	selu	5	0.15	895
8	85.17	True	True	8	relu	10	0.08	806
5	85.05	False	True	8	elu	3	0.18	563




Hyperparameter Importances
batch_size
max_dropout_rate
hidden_layers
max_hidden_units
hidden_activation
0.29
0.29
0.18
0.15
0.09
Hyperparameter
0
0.05
0.1
0.15
0.2
0.25
0.3
Importance for Objective Value


Parallel Coordinate Plot
87.3974
87
86.5
86
85.5
85.0528
Objective Value
64
32
16
8
batch_size
elu
selu
swish
relu
hidden_activation
10
1
hidden_layers
0.58396
0.00739
max_dropout_rate
980
130
max_hidden_units
Objective Value
87
86.5
86
85.5



7.6 Conclusioni

Dall'analisi di questo dataset si è potuto riscontrare che lo sbilanciamento delle classi è un fattore negativo per l'addestramento dei modelli. Questo problema persiste anche con l'utilizzo di operazioni di feature selection e class balancing effettuato con il framework Optuna.
Tuttavia, si è potuto riscontrare un leggero miglioramento nell'*accuracy*, anche se rimane basso il valore di precision, quindi alto il valore dei falsi positivi individuati.

Secondo il nostro ragionamento un buon sito di e-commerce dovrebbe sottostimare la quantità di vendite, per avere una previsione più cauta, soprattutto in termini di potenziali incassi.