Zbiór danych pobrany ze strony kaggle: link
Zbiór danych istnieje na stronie Machine Learning Repository(dalej MLR): link, wcześniej istniał na tej stronie pod inną nazwą.

Wybrałam stronę kaggle do importowania tego zbioru danych, ponieważ po aktualizacji linka na stronie MLR nie zawierał nazwę kolumn, a poprzedni plik z oryginałem został usunięty, ale został na stronie kaggle.

Zbiór danych zawiera dane kategoryczne:

- **work-class**: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked;
- **education**: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool;
- **marital-status**: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse;
- **occupation**: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces;
- **relationship**: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- **race:** White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black;
- **sex:** Female, Male;
- **native-country**: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holland-Netherlands.

Oraz dane liczbowe:

- **age**,
- **capital-gain**,
- **capital-loss**,
- **hours-per-week**,
- **salary**: <=50K / >50K .

Celem tego projektu jest przewidywanie wynagrodzenia danej osoby na podstawie jej cech osobistych, takich jak płeć, wykształcenie i kraj zamieszkania. Analizując te dane, staramy się zidentyfikować wzorce i trendy, które można wykorzystać do stworzenia modelu, który może dokładnie przewidzieć wynagrodzenie.

Projekt będzie obejmował użycie Random Forest Classifier oraz Logistyczną Regresję do zbudowania modeli oraz użycie GridSearchCV do znalezienia najlepszych hiperparametrów dla modeli. Wydajność modelu zostanie oceniona przy użyciu różnych metryk po czym zostanie wybrany najlepszy model.

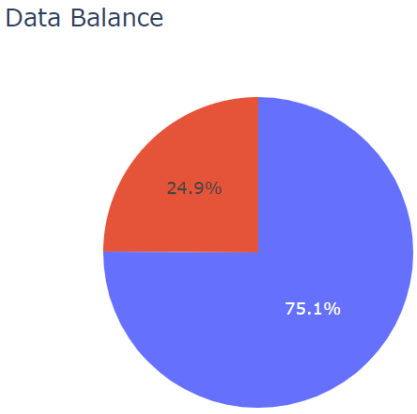
Na początku przeprowadzono czyszczenie danych: usunięte wiersze, zawierające brakujące wartości (w tym przypadku brakujące wartości były zaznaczone jako ‘ ?’) oraz usunięta columna “fnlwgt”, która nie jest potrzebna do analizy.

Analiza danych i przygotowanie danych

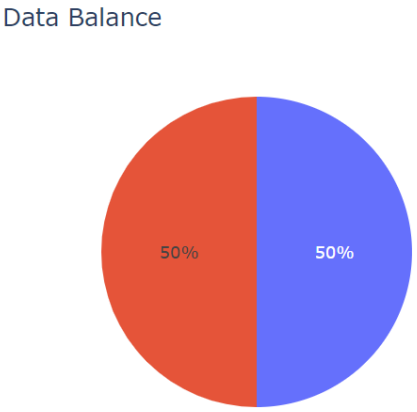
Przeprowadzono analizę danych za pomocą wizualizacji. To pokazało, że mamy do czynienia ze zbiorem niezbalansowanym a także, że między zmiennymi występują wysokie współczynniki korelacji, co wskazuje, że zbiór danych ma strukturę liniową.

Zastosowano algorytm generowania sztucznych przykładów SMOTE (ang. Synthetic Minority Over-Sampling Technique). Oversampling odbywa się poprzez tworzenie syntetycznych próbek klasy mniejszościowej. Parametr random_state służy do ustawienia ziarna (seed) generatora liczb losowych, tak aby wyniki mogły zostać odtworzone. Metoda fit_resample() stosuje SMOTE do danych wejściowych (X i y) i zwraca ponownie próbkowane dane (X_resampled i y_resampled).

Before balancing



After balancing



Dalej ponownie próbkowane dane (X_resampled, y_resampled) są podzielone na zestawy treningowe i testowe za pomocą funkcji train_test_split. Parametr test_size określa proporcję danych, które mają być użyte do testowania, w tym przypadku jest to 0,2 (lub 20%) danych. Pozostałe 80% danych posłużą do uczenia. Funkcja zwraca cztery zmienne: X_train i y_train, które są zbiorem uczącym odpowiednio dla cech i etykiet, X_test i y_test, które są zbiorami testowymi. Ten podział pozwala ocenić wydajność modelu na niewidocznych danych i uniknąć overfitting.

Pipelines

W kolejnym kroku stworzono dwa pipelines, jeden dla losowego klasyfikatora lasu (pipeline_rfc) i jeden dla regresji logistycznej (pipeline_lr). Każdy pipeline zawiera dwa kroki:

- StandardScaler(): standaryzuje funkcje, usuwając średnią i skalując do wariancji jednostkowej;
- RandomForestClassifier() i LogisticRegression(solver='liblinear'): Są to dwa klasyfikatory używane w pipelines.

Wybrano solver ‘liblinear’ po to, aby w dalszych krokach poszukiwania najlepszych hyperparametrów można było sprawdzić jaki z parametrów regularyzacji regresji logistycznej warto wykorzystać do naszego modelu. Także dany solver dobrze nadaje się do małych lub średnich zbiorów danych, jak w tym przypadku.
Metoda fit() jest następnie używana do trenowania potoku na danych treningowych (X_train i y_train). Metoda score() służy do oceny wydajności potoku na danych testowych (X_test i y_test).

Dalej przeprowadzono k-krotną walidację krzyżową na dwóch pipelines (pipeline_rfc i pipeline_lr), która służy do oszacowania wydajności modelu na niewidocznych danych. Funkcja przyjmuje jako dane wejściowe pipeline, cechę i dane docelowe oraz liczbę ‘cv’ w tym przypadku 5. Dalej wyświetlana jest średnia, która zapewnia oszacowanie wydajności modelu oraz odchylenie standardowe, które daje informację o zmienności oszacowanej wydajności.

```
scores = cross_val_score(pipeline_rfc, X_resampled, y_resampled, cv=5)
print(f'Accuracy: {scores.mean():.2f} +/- ({scores.std() * 2:.2f})')
```

```
Accuracy: 0.88 +/- (0.05)
```

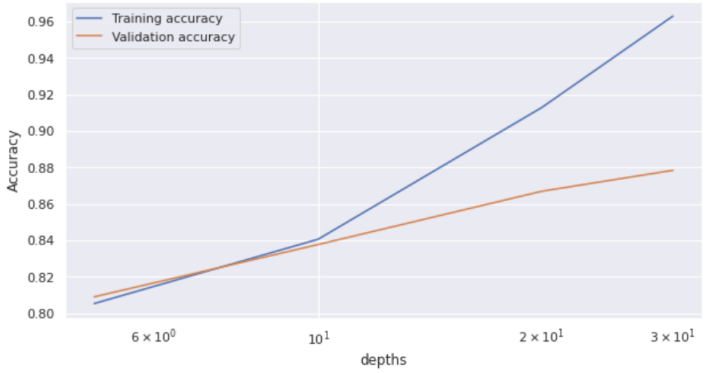
```
scores = cross_val_score(pipeline_lr, X_resampled, y_resampled, cv=5)
print(f'Accuracy: {scores.mean():.2f} +/- ({scores.std() * 2:.2f})')
```

```
Accuracy: 0.85 +/- (0.05)
```

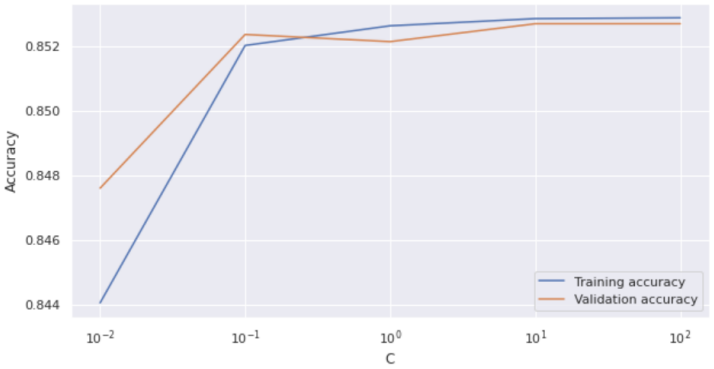
Następnie zastosowano funkcję learning_curve() do stworzenia wykresów pokazujących, jak zmienia się wydajność modelu wraz ze wzrostem liczby przykładów szkoleniowych. W tym przypadku tworzono krzywe uczenia się, zarówno dla losowego klasyfikatora lasu (pipeline_rfc), jak i klasyfikatora regresji logistycznej (pipeline_lr) przy użyciu zestawu danych (X_resampled, y_resampled). Używano 5-krotnej i 10-krotnej walidacji krzyżowej w funkcji i oceniano dokładność modelu. Uzyskane wykresy przedstawiają średnie wyniki dokładności szkolenia i walidacji jako funkcję liczby przykładów treningu. Z wykresów wynika, że oba modele wykazują dobrą wydajność przy rosnącej liczbie przykładów szkoleniowych, jednak klasyfikator losowego lasu wykazuje lepszą wydajność, ponieważ ma wyższą dokładność w zbiorze walidacyjnym.

Zwizualizowano dokładność modeli losowego klasyfikatora lasu (RFC) i regresji logistycznej (LR) jako funkcję hiperparametrów max_depth i C odpowiednio. Najpierw dzielono dane na zestawy treningowe i walidacyjne. Następnie wybiera różne wartości max_depth dla modelu RFC i C dla modelu LR i rejestruje dokładność szkolenia i walidacji dla każdej wartości. Na koniec wykreśla dokładność uczenia się i sprawdza dokładność dla każdego modelu w odniesieniu do odpowiednich hiperparametrów.

RFC (max_depth)

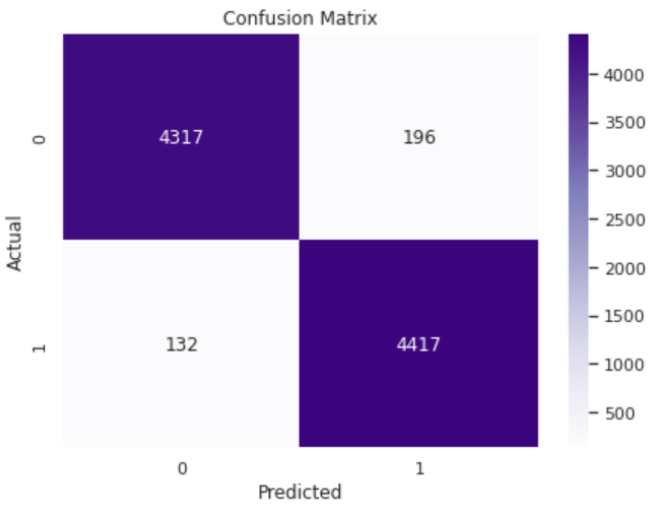


LR (C)

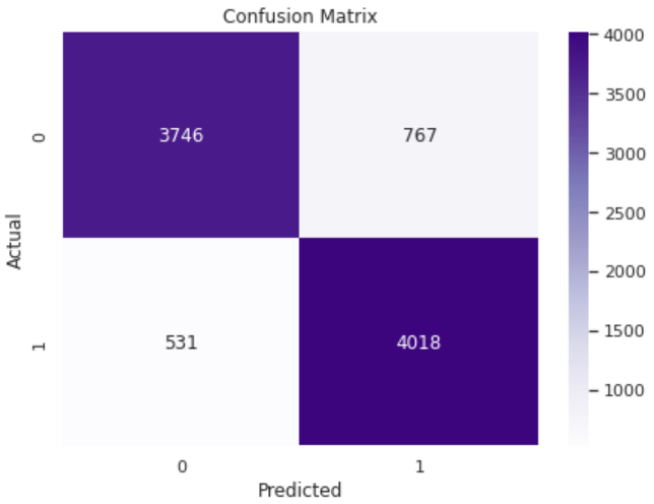


Następnie oceniano wydajność modeli pipeline_rfc i pipeline_lr przy użyciu macierzy pomyłek, precision, recall oraz f1.

RFC



LR



Poszukiwanie najlepszych hyperparametrów

Dalej zdefiniowano klasę o nazwie PipelineEvaluator, która pobiera pipeline i zestaw parametrów jako dane wejściowe i używa narzędzia GridSearchCV do przeprowadzenia zweryfikowanego krzyżowo wyszukiwania parametrów potoku w siatce. Następnie używamy klasy PipelineEvaluator dla pipeline_rfc oraz dla pipeline_lr. Zdefiniowano siatki parametrów dla każdego pipeline'u i wyświetlane są najlepsze parametry dla danych modeli. Wyniki testu są drukowane wraz ze średnią i odchyleniem standardowym wyników walidacji krzyżowej uzyskanych podczas poszukiwania siatki.

RFC

```
param_grid_rfc = {
    'classifier__n_estimators': [10, 100, 200],
    'classifier__max_depth': [10, 20, 30, 40],
    'classifier__criterion': ['gini', 'entropy']
}
```

```
evaluator = PipelineEvaluator(pipeline_rfc, param_grid_rfc)
score_rfc = evaluator.evaluate(X_train, y_train, X_test, y_test)
print(f'Cross-validation accuracy is: {evaluator.scores_mean:.3f} \
+/- {evaluator.scores_std:.3f}')
```

```
Cross-validation accuracy is: 0.858 +/- 0.001
```

```
best_params_rfc = evaluator.best_params_
best_params_rfc
```

```
{'classifier__criterion': 'entropy',
 'classifier__max_depth': 40,
 'classifier__n_estimators': 200}
```

LR

```
param_grid_lr = {
    'classifier__C': [0.001, 0.01, 0.1, 1, 10, 100],
    'classifier__penalty': ['l1', 'l2']
}
```

```
evaluator = PipelineEvaluator(pipeline_lr, param_grid_lr)
score_lr = evaluator.evaluate(X_train, y_train, X_test, y_test)
print(f'Cross-validation accuracy is: {evaluator.scores_mean:.3f} \
+/- {evaluator.scores_std:.3f}')
```

```
Cross-validation accuracy is: 0.841 +/- 0.001
```

```
best_params_lr = evaluator.best_params_
best_params_lr
```

```
{'classifier__C': 1, 'classifier__penalty': 'l2'}
```

Ewaluacja modeli w GridSearchCV

Stworzono funkcje make_scorer który wykorzystano do oceniania modeli za pomocą różnych metryk. Użyto argumentu pos_label do określenia pozytywnej etykiety klasy i argument great_is_better do określenia, że wyższy wynik jest lepszy.

Obiekt GridSearchCV jest następnie dopasowywany przy użyciu danych X_train i y_train, po czym wyświetlane są najlepsze parametry i wyniki. Metoda predict_proba jest używana do uzyskania przewidywanych prawdopodobieństw dla zestawu testowego, a te prawdopodobieństwa są następnie wykorzystywane do obliczenia współczynnika wyników fałszywie dodatnich (false positive rate) i współczynnika wyników prawdziwie dodatnich (true positive rate) dla krzywej ROC. Obliczany jest również obszar pod krzywą ROC (AUC), który jest powszechną miarą wydajności klasyfikatorów binarnych.

Strojenie wybranego modelu

Na podstawie uzyskanych wyników najlepszym wybranym modelem jest Random Forest Classifier o parametrach: n_estimators=200, max_depth=40, criterion='entropy' oraz wykorzystujący StandardScaler() do skalowania cech. Ten model osiągnął wynik F1 na poziomie 0.964%, oraz wynik ROC AUC na poziomie 0.963%. Oznacza to, że model jest w stanie dokładnie przewidzieć zmienną docelową z wysokim poziomem precision i recall. Wynik ROC AUC potwierdza również, że model jest w stanie dobrze rozróżnić różne klasy.

Macierz pomyłek do ostatecznego modelu wygląda następująco: