

ccp

May 6, 2024

## 1 CUSTOMER CHURN PREDICTION

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
import tkinter as tk
import sklearn
from tkinter import filedialog
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

```
[2]: data = pd.read_csv(r'C:/Users/gagan/Downloads/archive (2)/Churn_Modelling.csv')
```

```
[3]: data.head(5)
```

```
[3]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	
3	4	15701354	Boni	699	France	Female	39	
4	5	15737888	Mitchell	850	Spain	Female	43	

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1	1	
1	1	83807.86	1	0	1	
2	8	159660.80	3	1	0	
3	1	0.00	2	0	0	
4	2	125510.82	1	1	1	

	EstimatedSalary	Exited
0	101348.88	1

1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0

```
[4]: data = data.drop('CustomerId', axis=1)
```

```
[5]: data.dtypes
```

```
[5]: RowNumber      int64
Surname          object
CreditScore      int64
Geography        object
Gender           object
Age              int64
Tenure           int64
Balance          float64
NumOfProducts   int64
HasCrCard        int64
IsActiveMember   int64
EstimatedSalary float64
Exited           int64
dtype: object
```

```
[6]: data.isnull().sum()
```

```
[6]: RowNumber      0
Surname           0
CreditScore       0
Geography         0
Gender            0
Age               0
Tenure            0
Balance           0
NumOfProducts     0
HasCrCard         0
IsActiveMember    0
EstimatedSalary   0
Exited            0
dtype: int64
```

```
[7]: for column in data:
      print(column, data[column].unique())
```

```
RowNumber [    1     2     3 ... 9998 9999 10000]
Surname   ['Hargrave' 'Hill' 'Onio' ... 'Kashiwagi' 'Aldridge' 'Burbidge']
CreditScore [619 608 502 699 850 645 822 376 501 684 528 497 476 549 635 616 653
```

587

```
726 732 636 510 669 846 577 756 571 574 411 591 533 553 520 722 475 490
804 582 472 465 556 834 660 776 829 637 550 698 585 788 655 601 656 725
511 614 742 687 555 603 751 581 735 661 675 738 813 657 604 519 664 678
757 416 665 777 543 506 493 652 750 729 646 647 808 524 769 730 515 773
814 710 413 623 670 622 785 605 479 685 538 562 721 628 668 828 674 625
432 770 758 795 686 789 589 461 584 579 663 682 793 691 485 650 754 535
716 539 706 586 631 717 800 683 704 615 667 484 480 578 512 606 597 778
514 525 715 580 807 521 759 516 711 618 643 671 689 620 676 572 695 592
567 694 547 594 673 610 767 763 712 703 662 659 523 772 545 634 739 771
681 544 696 766 727 693 557 531 498 651 791 733 811 707 714 782 775 799
602 744 588 747 583 627 731 629 438 642 806 474 559 429 680 749 734 644
626 649 805 718 840 630 654 762 568 613 522 737 648 443 640 540 460 593
801 611 802 745 483 690 492 709 705 560 752 701 537 487 596 702 486 724
548 464 790 534 748 494 590 468 509 818 816 536 753 774 621 569 658 798
641 542 692 639 765 570 638 599 632 779 527 564 833 504 842 508 417 598
741 607 761 848 546 439 755 760 526 713 700 666 566 495 688 612 477 427
839 819 720 459 503 624 529 563 482 796 445 746 786 554 672 787 499 844
450 815 838 803 736 633 600 679 517 792 743 488 421 841 708 507 505 456
435 561 518 565 728 784 552 609 764 697 723 551 444 719 496 541 830 812
677 420 595 617 809 500 826 434 513 478 797 363 399 463 780 452 575 837
794 824 428 823 781 849 489 431 457 768 831 359 820 573 576 558 817 449
440 415 821 530 350 446 425 740 481 783 358 845 451 458 469 423 404 836
473 835 466 491 351 827 843 365 532 414 453 471 401 810 832 470 447 422
825 430 436 426 408 847 418 437 410 454 407 455 462 386 405 383 395 467
433 442 424 448 441 367 412 382 373 419]
Geography ['France' 'Spain' 'Germany']
Gender ['Female' 'Male']
Age [42 41 39 43 44 50 29 27 31 24 34 25 35 45 58 32 38 46 36 33 40 51 61 49
 37 19 66 56 26 21 55 75 22 30 28 65 48 52 57 73 47 54 72 20 67 79 62 53
 80 59 68 23 60 70 63 64 18 82 69 74 71 76 77 88 85 84 78 81 92 83]
Tenure [ 2  1  8  7  4  6  3 10  5  9  0]
Balance [      0.      83807.86 159660.8 ...  57369.61  75075.31 130142.79]
NumOfProducts [1 3 2 4]
HasCrCard [1 0]
IsActiveMember [1 0]
EstimatedSalary [101348.88 112542.58 113931.57 ...  42085.58  92888.52
38190.78]
Exited [1 0]
```

```
[8]: for column in data:
      if len(data[column].unique()) == 2:
          le = LabelEncoder()
          data[column] = le.fit_transform(data[column])
```

```
[9]: x = data.drop('Exited', axis=1)
      y = data['Exited']
```

```
numerical_columns = ['CreditScore', 'Age', 'Tenure', 'Balance',
                    'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
x = x[numerical_columns]
print(x.head(5))
for i in x:
    print(i)
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3)
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	\
0	619	42	2	0.00	1	1	
1	608	41	1	83807.86	1	0	
2	502	42	8	159660.80	3	1	
3	699	39	1	0.00	2	0	
4	850	43	2	125510.82	1	1	

	IsActiveMember	EstimatedSalary
0	1	101348.88
1	1	112542.58
2	0	113931.57
3	0	93826.63
4	1	79084.10

CreditScore

Age

Tenure

Balance

NumOfProducts

HasCrCard

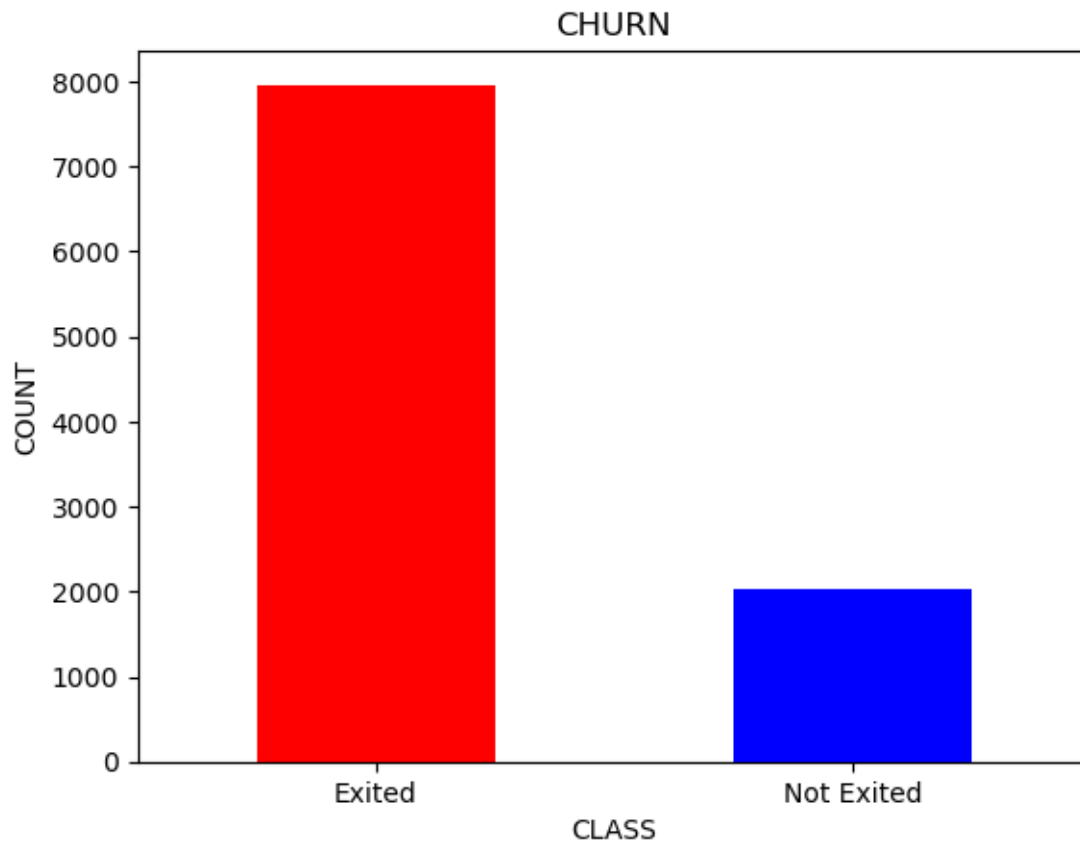
IsActiveMember

EstimatedSalary

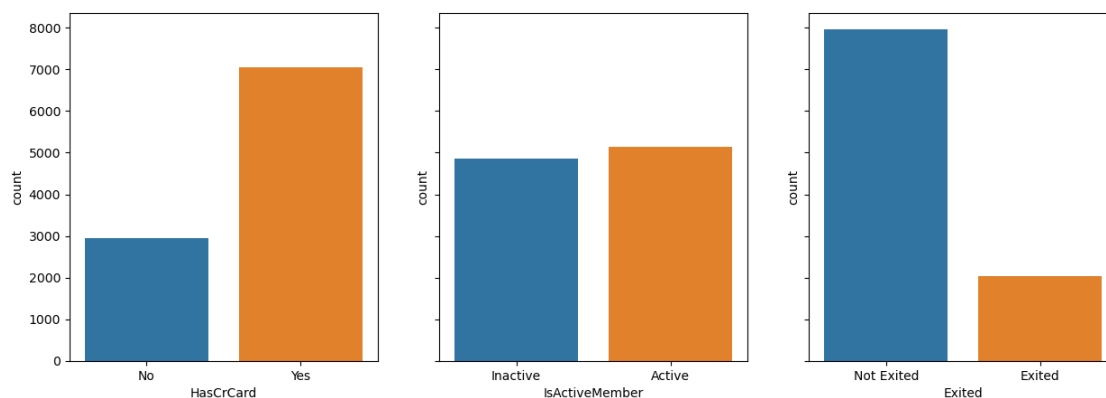
```
[10]: data.shape, xtrain.shape, xtest.shape
```

```
[10]: ((10000, 13), (7000, 8), (3000, 8))
```

```
[11]: LABELS = ["Exited", "Not Exited"]
COLORS = ["red", "blue"]
count_classes = pd.Series(data['Exited']).value_counts(sort=True)
count_classes.plot(kind='bar', rot=0, color=COLORS)
plt.title("CHURN")
plt.xticks(range(2), LABELS)
plt.xlabel("CLASS")
plt.ylabel("COUNT")
plt.show()
```



```
[12]: fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)
sns.countplot(x="HasCrCard", data=data, ax=axes[0])
sns.countplot(x="IsActiveMember", data=data, ax=axes[1])
sns.countplot(x="Exited", data=data, ax=axes[2])
axes[0].set_xticklabels(["No", "Yes"])
axes[1].set_xticklabels(["Inactive", "Active"])
axes[2].set_xticklabels(["Not Exited", "Exited"])
plt.show()
```

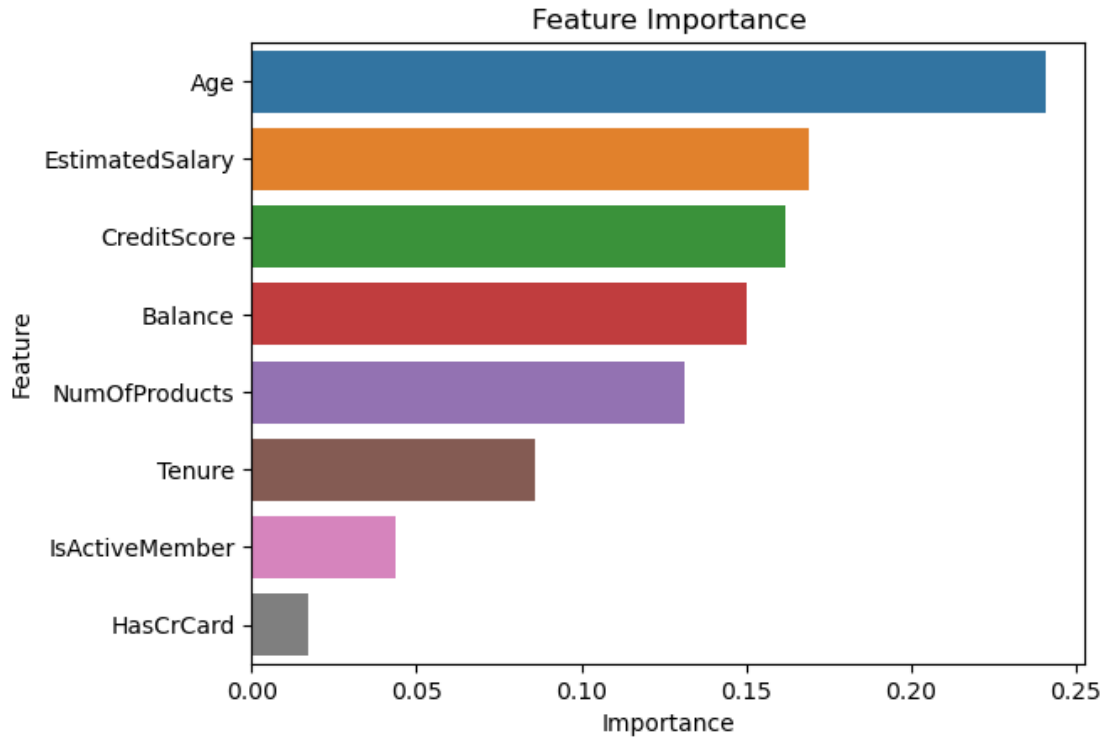


```
[13]: model = RandomForestClassifier()
model.fit(xtrain, ytrain)
accuracy = model.score(xtest, ytest)
print(f"Random Forest Accuracy: {round(accuracy * 100, 2)}%")
```

Random Forest Accuracy: 85.43%

```
[14]: feature_importances = pd.DataFrame({'Feature': xtrain.columns, 'Importance':
↳model.feature_importances_})
print(feature_importances.sort_values(by='Importance', ascending=True))
sns.barplot(x='Importance', y='Feature', data=feature_importances.
↳sort_values(by='Importance', ascending=False))
plt.title('Feature Importance')
plt.show()
```

	Feature	Importance
5	HasCrCard	0.016969
6	IsActiveMember	0.043607
2	Tenure	0.086101
4	NumOfProducts	0.131425
3	Balance	0.150242
0	CreditScore	0.161880
7	EstimatedSalary	0.168908
1	Age	0.240869



```
[15]: def predict_customer_churn(input_data):
    input_data = input_data[numerical_columns]
    input_data_scaled = sklearn.preprocessing.StandardScaler().
    ↪fit_transform(input_data)
    predictions = model.predict(input_data_scaled)
    return predictions

def plot_comparison(actual_exited, predicted_exited):
    actual_counts = np.bincount(actual_exited).tolist()
    predicted_counts = np.bincount(predicted_exited).tolist()
    labels = ['Not Exited', 'Exited']
    x = range(len(labels))
    plt.figure(figsize=(8, 6))
    plt.bar(x, actual_counts, width=0.4, align='center', label='Actual', ↪
    ↪alpha=0.8)
    plt.bar(x, predicted_counts, width=0.4, align='edge', label='Predicted', ↪
    ↪alpha=0.8)
    plt.xlabel('Exit Status')
    plt.ylabel('Count')
    plt.title('Comparison of Actual vs Predicted Exited')
    plt.xticks(x, labels)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

```

def get_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        selected_file_label.config(text="Selected file: " + file_path)
        global ted
        ted = pd.read_csv(file_path)
        print("File loaded into 'ted' variable.")
        ted2 = ted
        numerical_columns = ['CreditScore', 'Age', 'Tenure', 'Balance', '
↳ NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
        ted = ted[numerical_columns]
        ted.columns = numerical_columns
        predictions = predict_customer_churn(ted)
        for i in predictions:
            if i==1:
                print(f"Exited {i}")
            else:
                print(f"Not Exited {i}")
        predictions_proba = predict_customer_churn(ted)
        actual_churn = ted2['Exited']
        plot_comparison(actual_churn, predictions)

root = tk.Tk()
button = tk.Button(root, text="Select your test file", command=get_file)
button.pack(pady=100)
selected_file_label = tk.Label(root, text="Selected file: None")
selected_file_label.pack()
root.geometry("300x300")
root.mainloop()

```

File loaded into 'ted' variable.

Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1  
 Exited 1

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names

warnings.warn(

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X



does not have valid feature names, but RandomForestClassifier was fitted with feature names  
warnings.warn(

