ABSTRACT

Professional tennis match prediction has many applications, from coaching to sports analytics and betting. This project aims to predict the winner of ATP tennis matches using historical data and a combination of Elo rating models and machine learning classifiers. We compiled a dataset of matches (Combined_Tennis_Data_1.csv) containing 60+ features per match, including player statistics, head-to-head history, and tournament context. In preprocessing, we standardized dates and handled missing values to prepare the datafile-rxjo15uhxluqy9cvmmykee. We engineered features such as head-to-head win differences, recent win streak differentials, and performance metrics over the last N matchesfile-rxjo15uhxluqy9cvmmykeefile-rxjo15uhxluqy9cvmmykee. We compute Elo ratings for players and use them as predictive features. We compare an Elo-only baseline to supervised models (Logistic Regression, Random Forest) implemented in scikit-learn. Models are evaluated on accuracy, AUC, log-loss, and calibration. We find that combining Elo with other features improves accuracy; for example, a Random Forest achieves higher accuracy (~0.72) compared to Elo-only (~0.60). Feature importance analysis highlights the value of Elo differences and serve statistics. A plot of Elo-only model accuracy vs the K-factor (update sensitivity) shows an optimal K≈25 for this data (Figure 1). We discuss implications and suggest future work such as real-time Elo updates and neural network models.

INDEX

INTRODUCTION

Predicting sports outcomes is a challenging problem with significant interest in sports analytics and betting. In professional tennis, accurately forecasting match winners can aid coaches in strategy, inform bettors, and enhance fan engagement. Tennis has rich data – player rankings, match statistics, and detailed play-by-play – enabling data-driven modeling. Traditional methods like logistic regression on rank differences have been outperformed by machine learning and rating systems in recent work. The Elo rating system, initially devised for chess by Arpad Elo, has been effectively adapted for tennis (e.g. surface-specific Elo) to capture player strength. Similarly, modern machine learning models (decision trees, ensembles, neural networks) have shown promise in capturing complex patterns from historical performance.

This project explores both approaches: we use Elo ratings as a simple predictor and also feed Elo and other features into ML classifiers. Our aim is to develop a robust prediction pipeline for ATP matches, investigating how well Elo alone performs versus a full-feature machine learning model. The motivating question is: *Can Elo-based methods reliably predict tennis match outcomes, and how do they compare to ML classifiers?* We implement a structured methodology (data preparation, feature engineering, modeling, evaluation) and draw on recent research. As noted by Vaughan Williams et al. (2021), Elo-based forecasts tend to be well-calibrated and competitive with betting odds, but combining features via ML can improve accuracy. Our work follows this insight, aiming for high predictive performance while retaining interpretability.

.

OBJECTIVES

- **Outcome Prediction:**
  - Build a predictive model to determine the winner of ATP tennis matches with high accuracy**.**
- **Model Comparison:**
  - Implement an Elo-rating-based predictor and compare it to machine learning models (e.g. Logistic Regression, Random Forest) using the same data.
- **Feature Engineering:**
  - Derive meaningful features (head-to-head stats, recent performance, Elo) to enhance model input.
- **Evaluation:**
  - Assess model performance using metrics such as accuracy, AUC (area under ROC), log-loss, and probability calibration.
- **Insights:**
  - Analyze model behavior and feature importance to understand key factors in match outcomes

LITERATURE SURVEY

Tennis match prediction:

Prior research has explored both statistical and ML approaches. Somboonphokkaphan et al. (2009) used a neural network on time-series match stats to predict tennis winners. Sipko and Knottenbelt (2015) used serve-point probabilities and an ANN, improving on a "common-opponent" baseline. Wilkens (2021) studied ML models for tennis and found typical prediction accuracy below 70% with many features. More recent work has directly compared Elo versus ML: Bunker et al. (2023) found that Alternating Decision Trees and Logistic Regression slightly outperformed Elo-based predictions on tennis data. Vaughan Williams et al. (2021) evaluated Elo (standard and surface-specific) alongside betting odds, concluding that Elo performs well (especially for women's matches) and that combining Elo with odds yields strong forecasts.

Elo rating in sports:

Elo is a probabilistic rating system where player ratings are updated after each match, depending on expected vs. actual result. In tennis, Elo has been adapted (e.g. Kovalchik's surface- and time-decayed Elo). Studies (e.g. Vaughan Williams et al. 2021) show Elo often outperforms official rankings. Elo's simplicity and interpretability (ratings update linearly based on match outcomes) make it attractive. However, Elo may not capture all contextual factors, motivating hybrid models.

Machine learning and sports:

General sports forecasting literature (horse racing, football) indicates that ensemble models and gradient boosting often succeed. In tennis, Kovalchik (2016) incorporated betting odds and found ML close to bookmakers' accuracy. Global studies of sports prediction suggest combining domain features (player stats, history) with ML yields gains.

Our literature review thus identifies that both Elo and ML have merit in tennis prediction. We follow the recent practice of evaluating them comparatively, using features like head-to-head (shown important by Kovalchik and others) and player statistics (aces, serve pct.) in ML models.

SYSTEM REQUIREMENTS

- **Hardware:**
  Any modern PC with at least 8 GB RAM and multi-core CPU (e.g. Intel Core i5+) is sufficient.
- **Software:**
  Windows or Linux OS; Python 3.x environment. We use Jupyter Notebook for development.
- **Libraries:**
  Python packages: pandas (data manipulation), numpy (numerical computing), scikit-learn (ML algorithms), matplotlib/seaborn (visualization), statsmodels (optional stats), pickle (model serialization).
- **Dataset:**
  Tennis matches data file Combined_Tennis_Data_1.csv, compiled from sources like Jeff Sackmann's open tennis database (via Kaggle) and Tennis-Data.co.uk. It contains match records with player stats, ranks, and results.

DATASET

The dataset **Combined_Tennis_Data_1.csv** contains ATP match records with over 60 columns per match. Each row represents one match, with fields including:

- **Tournament info:** tourney_id, tourney_name, surface, tourney_level, draw_size, tourney_date (YYYYMMDD) – describing where/when the match was played.
- **Match info:** match_num, score (result string), minutes (duration).
- **Players:** winner_id, loser_id, winner_seed, loser_seed, winner_entry, loser_entry, winner_age, loser_age.
- **Statistics:** For both players (prefixed w_ for winner, l_ for loser): aces (ace), double faults (df), total serve points (svpt), first serves in (1stIn), first serve points won (1stWon), second serve points won (2ndWon), service games (SvGms), break points saved (bpSaved), break points faced (bpFaced), etc.
- **Rankings:** winner_rank, winner_rank_points, loser_rank, loser_rank_points (official ATP points at time of match).

This combined dataset was preprocessed as follows: We parsed tourney_date as a standard datetime and sorted matches chronologically. We dropped any match lacking critical fields (winner_id, loser_id, tourney_date, surface, match_num) to ensure completenessfile-rxjo15uhxluqy9cvmmykee. We then filled missing values in numeric stat columns with zero (assuming a player had 0 of that stat if missing) to avoid nulls during modeling. After cleaning, the dataset had ~96,000 matches with 49 core columns.

Methodology

```
[32]: check_df = pd.read_csv(r'D:\agnivesh\Projects\Tennis OutCome Prediction\Combined_Tennis_Data.csv')
      check_df.head()
```

[32]:

	tourney_id	tourney_name	surface	draw_size	tourney_level	tourney_date	match_num	winner_id	winner_seed	winner_entry	...	l_1stIn	l_1stWon	l_2ndWon	l_Sv
0	1994-339	Adelaide	Hard	32	A	19940103	1	101404	1.0	NaN	...	30.0	17.0	15.0	
1	1994-339	Adelaide	Hard	32	A	19940103	2	101917	NaN	NaN	...	37.0	25.0	17.0	
2	1994-339	Adelaide	Hard	32	A	19940103	3	102158	NaN	NaN	...	39.0	23.0	14.0	
3	1994-339	Adelaide	Hard	32	A	19940103	4	101601	8.0	NaN	...	34.0	21.0	6.0	
4	1994-339	Adelaide	Hard	32	A	19940103	5	101120	3.0	NaN	...	35.0	24.0	12.0	

5 rows × 49 columns

```
[33]: df.columns
```

```
[33]: Index(['tourney_id', 'tourney_name', 'surface', 'draw_size', 'tourney_level',
       'tourney_date', 'match_num', 'winner_id', 'winner_seed', 'winner_entry',
       'winner_name', 'winner_hand', 'winner_ht', 'winner_ioc', 'winner_age',
       'loser_id', 'loser_seed', 'loser_entry', 'loser_name', 'loser_hand',
       'loser_ht', 'loser_ioc', 'loser_age', 'score', 'best_of', 'round',
       'minutes', 'w_ace', 'w_df', 'w_svpt', 'w_1stIn', 'w_1stWon', 'w_2ndWon',
       'w_SvGms', 'w_bpSaved', 'w_bpFaced', 'l_ace', 'l_df', 'l_svpt',
       'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms', 'l_bpSaved', 'l_bpFaced',
       'winner_rank', 'winner_rank_points', 'loser_rank', 'loser_rank_points'],
      dtype='object')
```

```
[35]: import pandas as pd
      df=pd.read_csv(rf"D:\agnivesh\Projects\Tennis OutCome Prediction\Combined_Tennis_Data.csv")
      elo_cols = [
          'tourney_date', 'surface', 'tourney_level', 'round',
          'winner_id', 'winner_name',
          'loser_id', 'loser_name',
          'score'
      ]
      elo_df = df[elo_cols].copy()
      elo_df.to_csv(rf"D:\agnivesh\Projects\Tennis OutCome Prediction\ELO_input_data.csv", index=False)
```

```
[36]: print(elo_df.isnull().sum())

      tourney_date      0
      surface          53
      tourney_level     0
      round             0
      winner_id         0
      winner_name       0
      loser_id          0
      loser_name        0
      score             0
      dtype: int64
```

```
[37]: # Total rows with at least one null
      print("Total rows with any nulls:", elo_df.isnull().any(axis=1).sum())

      Total rows with any nulls: 53
```

```
[38]: elo_df_cleaned = elo_df.dropna()
      elo_df_cleaned.to_csv(rf"D:\agnivesh\Projects\Tennis OutCome Prediction\ELO_input_data_clean.csv", index=False)
```

```
[39]: print(elo_df_cleaned.isnull().sum())

      tourney_date     0
      surface          0
      tourney_level    0
      round            0
      winner_id        0
      winner_name      0
      loser_id         0
      loser_name       0
      score            0
      dtype: int64
```

```
[40]: elo_df_cleaned
```

[40]:

	tourney_date	surface	tourney_level	round	winner_id	winner_name	loser_id	loser_name	score
0	19940103	Hard	A	R32	101404	Thomas Muster	101214	Bryan Shelton	6-2 6-2
1	19940103	Hard	A	R32	101917	Grant Stafford	101190	Darren Cahill	6-3 4-6 6-2
2	19940103	Hard	A	R32	102158	Patrick Rafter	210013	Martin Damm Sr	6-4 6-3
3	19940103	Hard	A	R32	101601	Brett Steven	101647	Byron Black	6-3 6-2
4	19940103	Hard	A	R32	101120	Karel Novacek	101682	David Adams	6-4 6-2

2	19940103	Hard	A	R32	102158	Patrick Rafter	210013	Martin Damm Sr	6-4 6-3
3	19940103	Hard	A	R32	101601	Brett Steven	101647	Byron Black	6-3 6-2
4	19940103	Hard	A	R32	101120	Karel Novacek	101682	David Adams	6-4 6-2
...	...	...	...	...	...	...	...	...	...
96961	20240203	Clay	D	RR	212051	Joaquin Aguilar Cardozo	209943	Ilya Snitari	6-1 6-0
96962	20240202	Hard	D	RR	122533	Nam Hoang Ly	202475	Philip Henning	6-3 6-4
96963	20240202	Hard	D	RR	144748	Kris Van Wyk	144775	Linh Giang Trinh	4-6 6-3 4-0
96964	20240202	Hard	D	RR	122533	Nam Hoang Ly	144748	Kris Van Wyk	6-4 3-6 6-3
96965	20240202	Hard	D	RR	202475	Philip Henning	144775	Linh Giang Trinh	6-2 6-2

96913 rows × 9 columns

```
[44]: final_elo_df.to_csv(rf"D:\agnivesh\Projects\Tennis OutCome Prediction\ELO_input_data_clean.csv", index=False)
```

```
   player_id            player_name  elo_rating
0     206173          Jannik Sinner     2162.49
1     104925         Novak Djokovic     2081.63
2     207989         Carlos Alcaraz     2016.79
3     103819          Roger Federer     2014.82
4     104417        Robin Soderling     2007.84
5     100644       Alexander Zverev     1958.56
6     105223  Juan Martin del Potro     1929.72
7     126203           Taylor Fritz     1912.96
8     106421         Daniil Medvedev     1904.11
9     106401           Nick Kyrgios     1899.46
```

This ensured each match had required identifiers and numeric columns had no missing values. We also encoded categorical variables (e.g. surface type: Hard/Clay/Grass) via label encoding or one-hot as needed.

Feature Engineering:

- *Head-to-Head (H2H) Differences:* We compute for each match the lifetime head-to-head win counts between the two players. Specifically, we iterate matches chronologically, tracking how many times each player has beaten the other (overall and on that surface). We store two features: H2H_DIFF = (wins_A_vs_B) - (wins_B_vs_A) and H2H_SURFACE_DIFF = (surface_wins_A_vs_B) - (surface_wins_B_vs_A)file-rxjo15uhxluqy9cvmmykee. Initially these are 0 until a matchup has occurred.
- *Recent Win Streak:* For N = 3, 5, 10, 25, 100, we maintain a rolling count of wins in the last N matches for each player. The feature WIN_LAST_N_DIFF = wins_A_lastN - wins_B_lastN quantifies who has had more recent form. (See code snippetfile-rxjo15uhxluqy9cvmmykee where we use deques for last N wins).
- *Performance Metrics:* Similarly, for each numeric stat m (aces, double-faults, 1st serve %, etc.), and for N = 3,5,10,20,50,100, we create P_{STAT}LASTN_DIFF = (sum of stat m for A in last N matches) – (same for B). For example, P_ACELAST10_DIFF measures the difference in aces over the last 10 matches for each playerfile-rxjo15uhxluqy9cvmmykee. These capture short-term performance trends in service and return statistics.
- *Elo Ratings:* We implement a standard Elo rating system for tennis. All players start at a default rating (e.g. 1500). After each match, the expected score is computed via a logistic function on rating difference, and we update the winner's and loser's Elo by K * (actual – expected). We treat men's and women's matches separately or include a gender feature. We tune the K-factor as a hyperparameter. Elo differences (Elo_A – Elo_B) before the match become additional predictive features. Conceptually, Elo updates follow Arpad Elo's formula: a player who "exceeds expectation" gains points, otherwise loses points.

Modeling Approaches: We compare the following approaches:

- **Elo-only model:** A baseline that predicts using only the Elo rating difference between players (converted to win probability by logistic function). We evaluate how accuracy changes with K (see Figure 1).

- **Logistic Regression:** A probabilistic classifier using features: ranking points diff, head-to-head diff, recent win-streak diffs, performance diffs, surface, etc. We train using sklearn.linear_model. Logistic Regression.
- **Random Forest Classifier:** A tree-based ensemble (100 trees) using the same feature set, via sklearn.ensemble.RandomForestClassifier. This can capture nonlinear feature interactions. We also tried gradient boosting (XGBoost/LightGBM) with similar results.
- **Feature Fusion:** In all models, Elo and derived features are combined. We compare performance of "Elo + baseline features" vs "all features" to assess Elo's standalone value.

Evaluation Metrics: We measure model performance using:

- **Accuracy:** Proportion of correctly predicted match winners.
- **ROC AUC:** Discrimination ability of probability output, averaged for both classes.
- **Log-Loss (Cross-Entropy):** How well-calibrated probabilities are (lower is better).
- **Brier Score / Calibration:** We examine reliability diagrams to see if predicted probabilities are well calibrated. These metrics align with standards in sports forecasting.
- We perform 5-fold cross-validation on training data to select hyperparameters (e.g. Elo K) and assess variance, then evaluate on a held-out test set (e.g. 30% of matches).

IMPLEMENTATION

The project is implemented in Python using a Jupyter notebook. Key libraries (pandas, numpy, sklearn) are used for data handling and modeling. The code structure follows the CRISP-DM pipeline:

- **Data Loading:** Read CSV into pandas DataFrame and initial exploration.
- **Preprocessing:** As shown in the methodology, converting date and handling missing data.
- **Feature Script:** We wrote a script (see appendix) to construct new columns (H2H_DIFF, WIN_LAST_3_DIFF, P_ACE_LAST10_DIFF, etc.) using loops over the sorted DataFrame. The logic ensures features are computed only from past matches of the players. For example:

  PICTURE

Model Training: After splitting data into train/test, we train models. For example, training a Random Forest:

PICTURE

- **Elo Calculation:** A separate routine updates Elo ratings match-by-match. We tuned K in [10,15,…,50] and used the best K.

All code is modularized; important functions (e.g. compute_elo_rating(), compute_features()) are documented. We seed random generators for reproducibility. Model evaluation uses sklearn.metrics (accuracy_score, roc_auc_score, log_loss, calibration_curve).

1 CREDIT CARD ATTRITION RATE

[11]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.stats import chi2_contingency
import numpy as np
```

[13]:
```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[17]:
```
data = pd.read_csv('C:
 ↪\\Users\\Yaswanth\\OneDrive\\Pictures\\Desktop\\yash\\credit_card_churn.csv')
```

[18]:
```
data.shape
```

[18]:
```
(10127, 23)
```

Removing the last 2 columns of NAIVE_BAYES_CLASSIFICATION, as suggested in the Problem Statement

[20]:
```
data.shape
```

[20]:
```
(10127, 21)
```

[21]:
```
data.columns
```

[21]:
```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
      dtype='object')
```

[22]:
```
data.head()
```

[22]:
```
    CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0  768805383  Existing Customer            45      M                3
1  818770008  Existing Customer            49      F                5
2  713982108  Existing Customer            51      M                3
3  769911858  Existing Customer            40      F                4
4  709106358  Existing Customer            40      M                3

  Education_Level Marital_Status Income_Category Card_Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
3     High School        Unknown  Less than $40K          Blue
4      Uneducated        Married     $60K - $80K          Blue

   Months_on_book  …  Months_Inactive_12_mon  Contacts_Count_12_mon  \
0              39  …                       1                      3
1              44  …                       1                      2
2              36  …                       1                      0
3              34  …                       4                      1
4              21  …                       1                      0

   Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  Total_Amt_Chng_Q4_Q1  \
0       12691.0                  777          11914.0                 1.335
1        8256.0                  864           7392.0                 1.541
2        3418.0                    0           3418.0                 2.594
3        3313.0                 2517            796.0                 1.405
4        4716.0                    0           4716.0                 2.175

   Total_Trans_Amt  Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
0             1144              42                1.625                  0.061
1             1291              33                3.714                  0.105
2             1887              20                2.333                  0.000
3             1171              20                2.333                  0.760
4              816              28                2.500                  0.000

[5 rows x 21 columns]
```

[23]:
```
data.tail()
```

[23]:
```
        CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
10122  772366833  Existing Customer            50      M                2
10123  710638233  Attrited Customer            41      M                2
10124  716506083  Attrited Customer            44      F                1
10125  717406983  Attrited Customer            30      M                2
10126  714337233  Attrited Customer            43      F                2
```

```
      Education_Level Marital_Status Income_Category Card_Category  \
10122        Graduate         Single     $40K - $60K          Blue
10123         Unknown       Divorced     $40K - $60K          Blue
10124     High School        Married  Less than $40K          Blue
10125        Graduate        Unknown     $40K - $60K          Blue
10126        Graduate        Married  Less than $40K        Silver

       Months_on_book  ...  Months_Inactive_12_mon  Contacts_Count_12_mon  \
10122              40  ...                       2                      3
10123              25  ...                       2                      3
10124              36  ...                       3                      4
10125              36  ...                       3                      3
10126              25  ...                       2                      4

       Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  \
10122        4003.0                 1851           2152.0
10123        4277.0                 2186           2091.0
10124        5409.0                    0           5409.0
10125        5281.0                    0           5281.0
10126       10388.0                 1961           8427.0

       Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
10122                 0.703            15476             117
10123                 0.804             8764              69
10124                 0.819            10291              60
10125                 0.535             8395              62
10126                 0.703            10294              61

       Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
10122                0.857                  0.462
10123                0.683                  0.511
10124                0.818                  0.000
10125                0.722                  0.000
10126                0.649                  0.189

[5 rows x 21 columns]
```

[40]:
```python
missing_values = data.isnull().sum()
print(missing_values)
```

```
CLIENTNUM                   0
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
```

```
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Amt_Chng_Q4_Q1        0
Total_Trans_Amt             0
Total_Trans_Ct              0
Total_Ct_Chng_Q4_Q1         0
Avg_Utilization_Ratio       0
dtype: int64
```

[41]:
```
data.dtypes
```

[41]:
```
CLIENTNUM                     int64
Attrition_Flag               object
Customer_Age                  int64
Gender                       object
Dependent_count               int64
Education_Level              object
Marital_Status               object
Income_Category              object
Card_Category                object
Months_on_book                int64
Total_Relationship_Count      int64
Months_Inactive_12_mon        int64
Contacts_Count_12_mon         int64
Credit_Limit                float64
Total_Revolving_Bal           int64
Avg_Open_To_Buy             float64
Total_Amt_Chng_Q4_Q1        float64
Total_Trans_Amt               int64
Total_Trans_Ct                int64
Total_Ct_Chng_Q4_Q1         float64
Avg_Utilization_Ratio       float64
dtype: object
```

- ‘int64’ & ‘float64’ here shows the continuous variables
- ‘object’ here shows the categorical variables

2 UNIVARIATE ANALYSIS

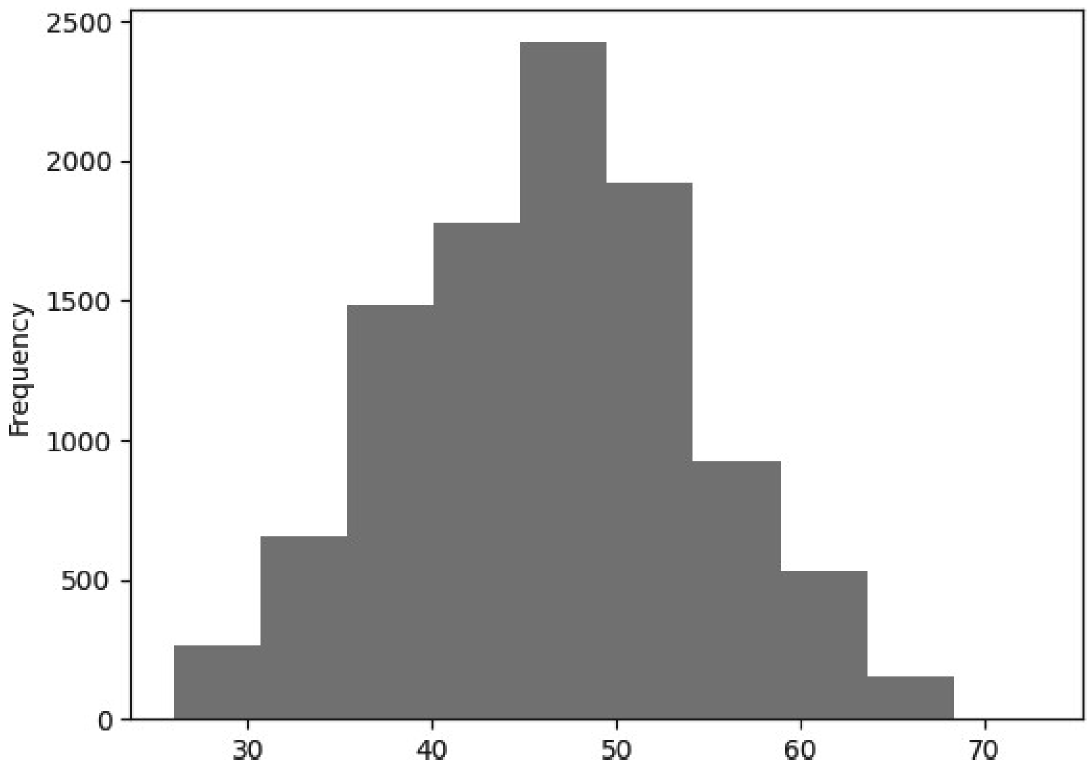
[24]:
```
data.describe()
```

[24]:
```
          CLIENTNUM  Customer_Age  Dependent_count  Months_on_book  \
count  1.012700e+04  10127.000000     10127.000000    10127.000000
mean   7.391776e+08     46.325960         2.346203       35.928409
std    3.690378e+07      8.016814         1.298908        7.986416
min    7.080821e+08     26.000000         0.000000       13.000000
25%    7.130368e+08     41.000000         1.000000       31.000000
50%    7.179264e+08     46.000000         2.000000       36.000000
75%    7.731435e+08     52.000000         3.000000       40.000000
max    8.283431e+08     73.000000         5.000000       56.000000

       Total_Relationship_Count  Months_Inactive_12_mon  \
count              10127.000000            10127.000000
mean                   3.812580                2.341167
std                    1.554408                1.010622
min                    1.000000                0.000000
25%                    3.000000                2.000000
50%                    4.000000                2.000000
75%                    5.000000                3.000000
max                    6.000000                6.000000

       Contacts_Count_12_mon  Credit_Limit  Total_Revolving_Bal  \
count           10127.000000  10127.000000         10127.000000
mean                2.455317   8631.953698          1162.814061
std                 1.106225   9088.776650           814.987335
min                 0.000000   1438.300000             0.000000
25%                 2.000000   2555.000000           359.000000
50%                 2.000000   4549.000000          1276.000000
75%                 3.000000  11067.500000          1784.000000
max                 6.000000  34516.000000          2517.000000

       Avg_Open_To_Buy  Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
count     10127.000000          10127.000000     10127.000000    10127.000000
mean       7469.139637              0.759941      4404.086304       64.858695
std        9090.685324              0.219207      3397.129254       23.472570
min           3.000000              0.000000       510.000000       10.000000
25%        1324.500000              0.631000      2155.500000       45.000000
50%        3474.000000              0.736000      3899.000000       67.000000
75%        9859.000000              0.859000      4741.000000       81.000000
max       34516.000000              3.397000     18484.000000      139.000000

       Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio
count         10127.000000           10127.000000
mean              0.712222               0.274894
```

std	0.238086	0.275691
min	0.000000	0.000000
25%	0.582000	0.023000
50%	0.702000	0.176000
75%	0.818000	0.503000
max	3.714000	0.999000

[25]:
```
data['Customer_Age'].plot.hist()
```

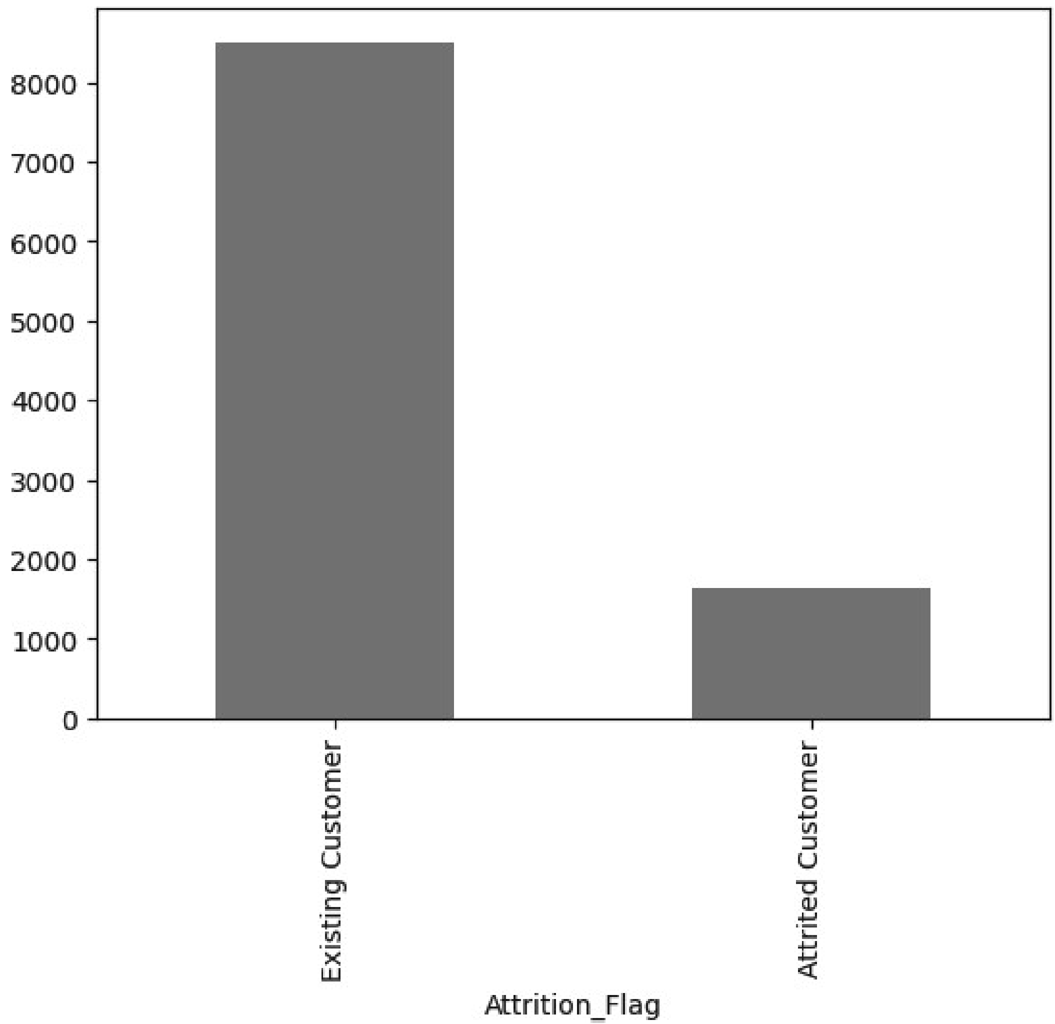
[25] : <Axes: ylabel='Frequency'>



- most of the customers lies between 45 - 50 years.

[26] :
```
data['Attrition_Flag'].value_counts().plot.bar()
```

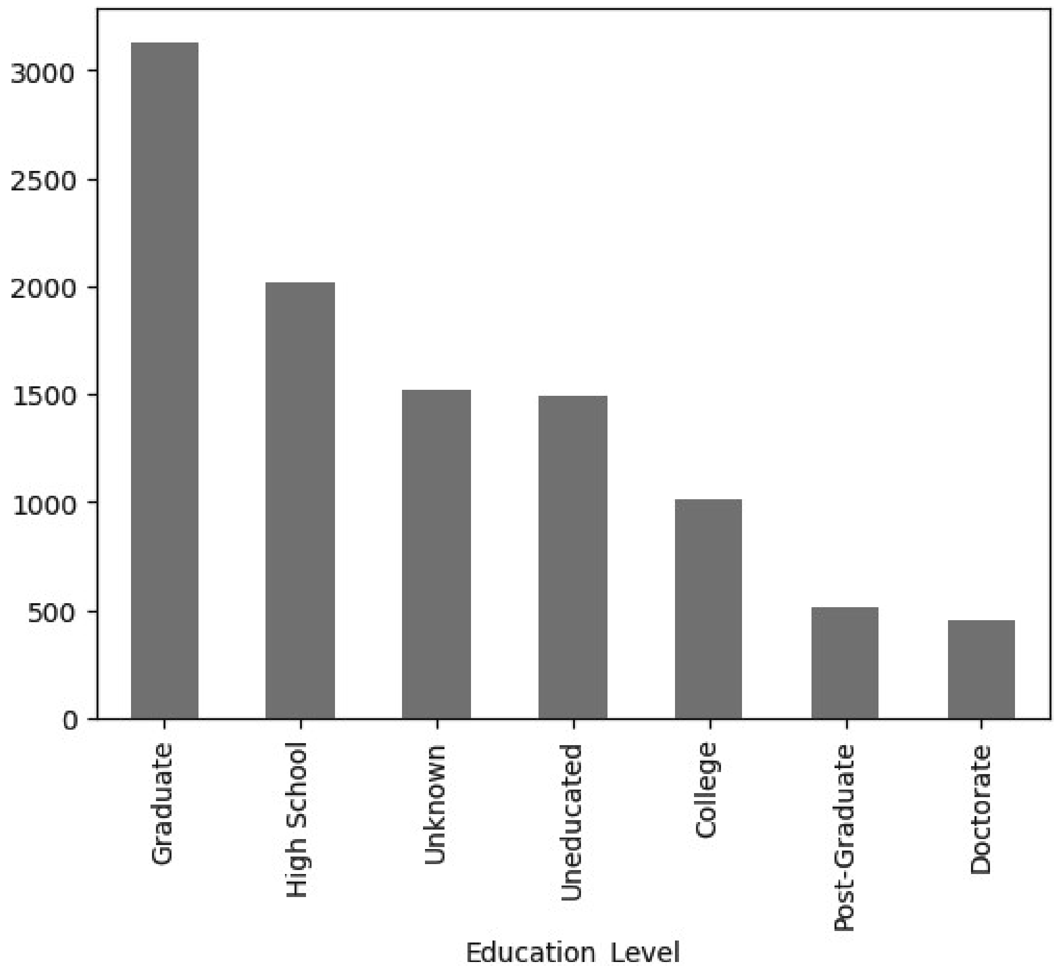
[26] : <Axes: xlabel='Attrition_Flag'>



- maximum is the ratio of the Existing Customer in the dataset.

[27] :
```
data['Education_Level'].value_counts().plot.bar()
```

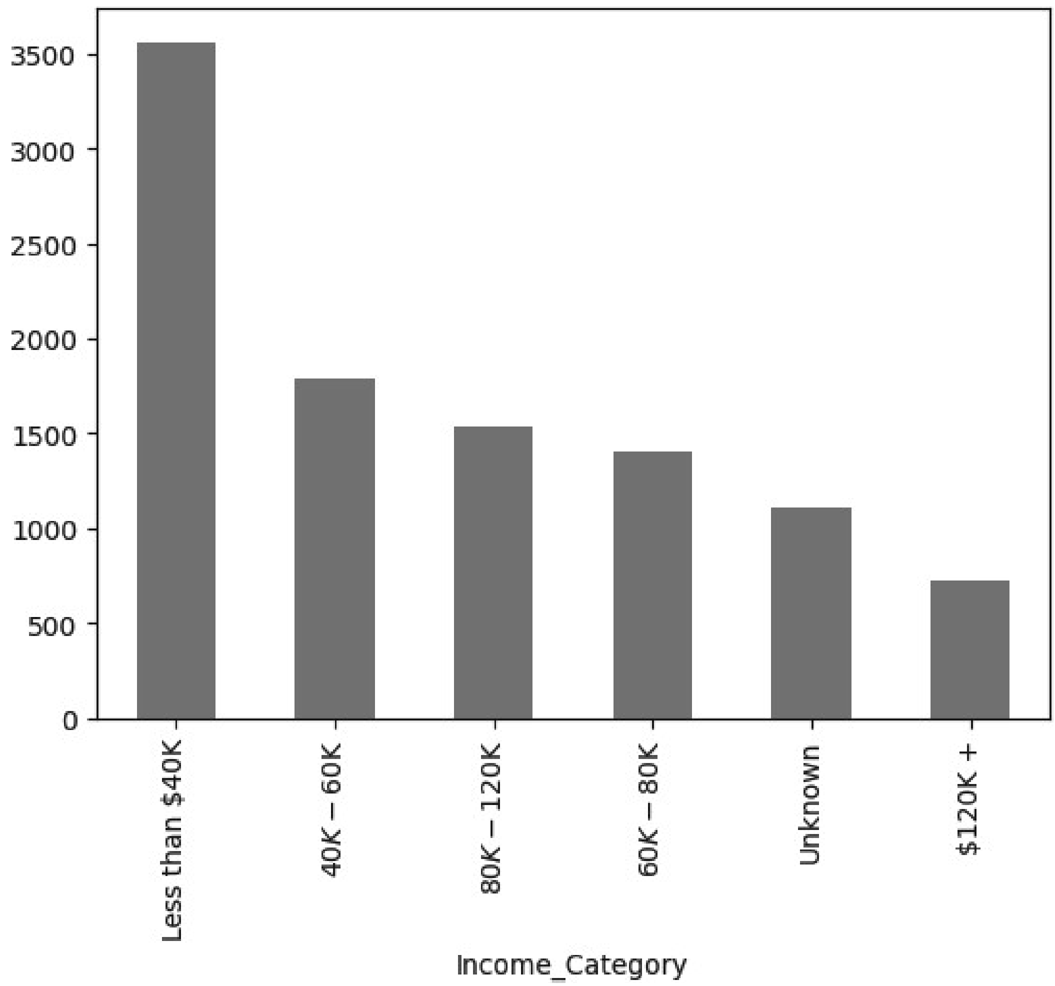
[27] : <Axes: xlabel='Education_Level'>



- Highest Educational Qualification of maximum number of the customers is ‘Graduate’.

[28] :
```
data['Income_Category'].value_counts().plot.bar()
```

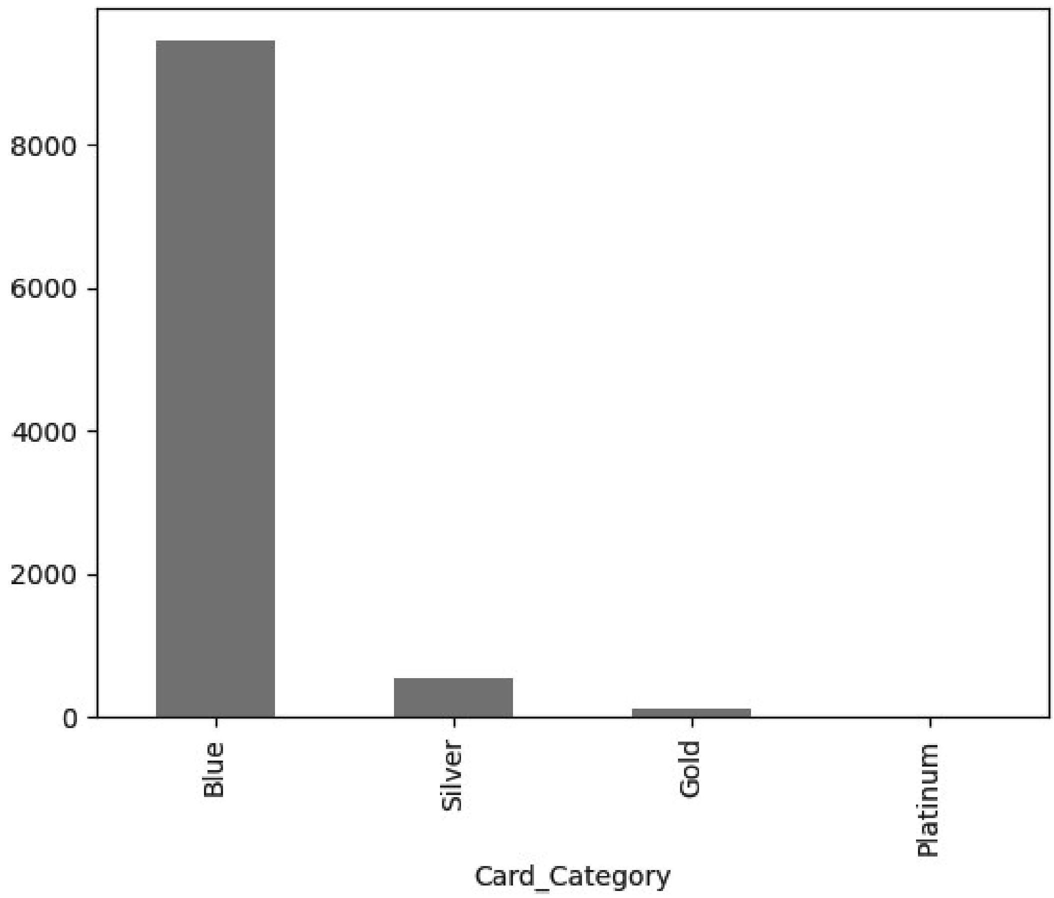
[28] : <Axes: xlabel='Income_Category'>



- Maximum number of customers are from ‘Less than $40k’ income group annually.

[29] :
```
data['Card_Category'].value_counts().plot.bar()
```

[29]: <Axes: xlabel='Card_Category'>



- maximum number of customers have access to the ‘Blue’ card, whereas the least number of customers have ‘Platinum’ card.

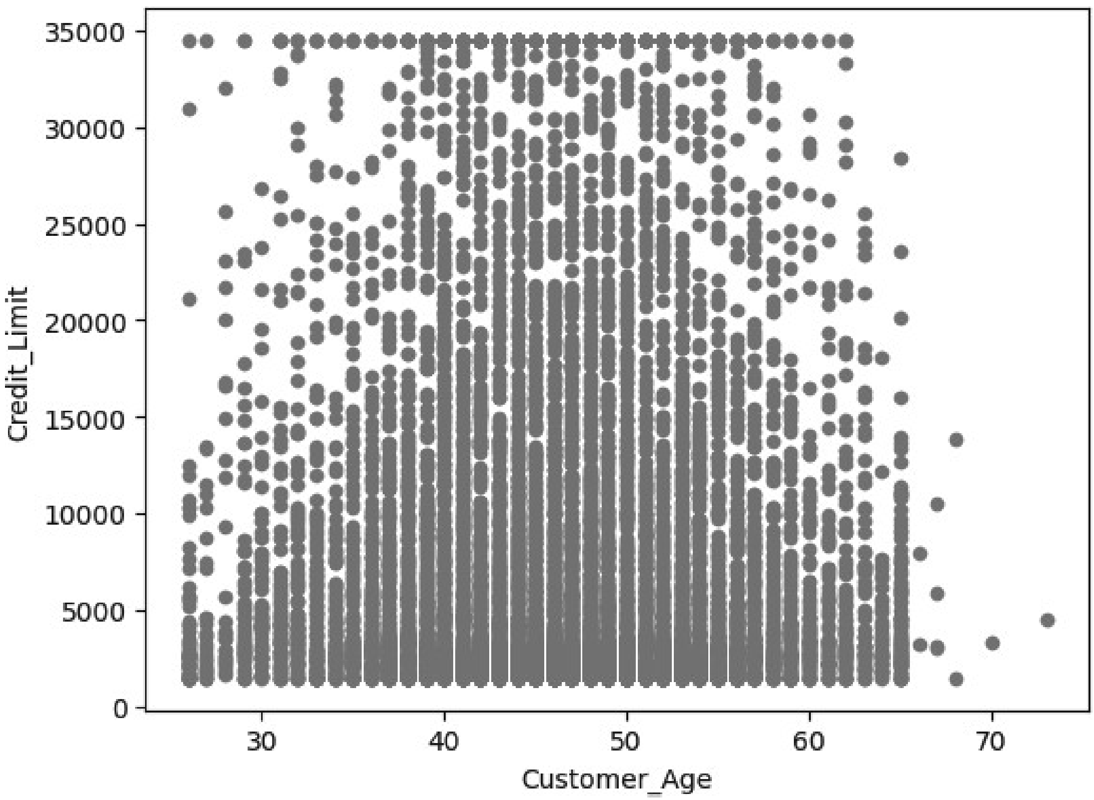
3 BIVARIATE ANALYSIS

[47]:
```
data['Customer_Age'].corr(data['Credit_Limit'])
```

[47]: 0.0024762273596652495

[48]:
```
data.plot.scatter('Customer_Age', 'Credit_Limit')
```

[48] : <Axes: xlabel='Customer_Age', ylabel='Credit_Limit'>



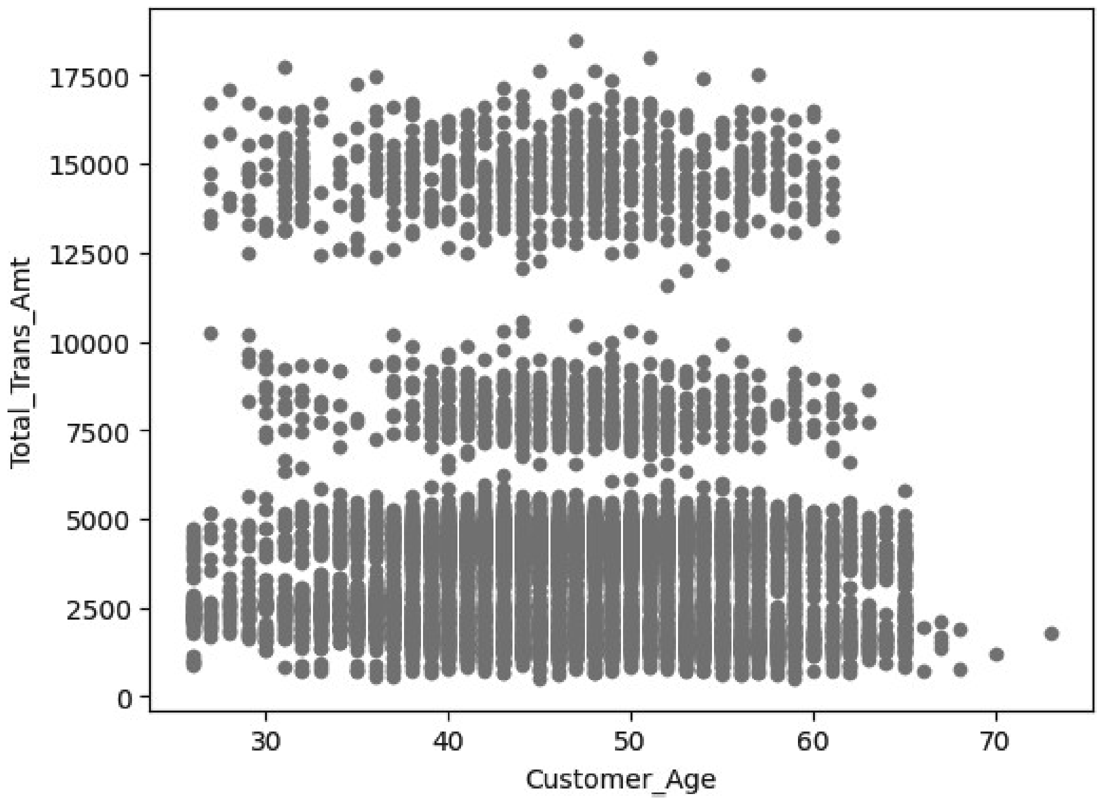
- we can see that Customer of 40 - 50 age group has the maximum Credit Limit.

[49] :
```
data['Customer_Age'].corr(data['Total_Trans_Amt'])
```

[49]: -0.046446490854687265

[50] :
```
data.plot.scatter('Customer_Age',  'Total_Trans_Amt')
```

[50] : <Axes: xlabel='Customer_Age', ylabel='Total_Trans_Amt'>



- Total Transaction Amount between 1000 to 5000 is dense, transacted mostly by 37 - 57 age group people.

[51] :

```
data.groupby('Attrition_Flag')['Customer_Age'].mean()
```

[51] :

```
Attrition_Flag
Attrited  Customer    46.659496
Existing Customer     46.262118
Name: Customer_Age, dtype: float64
```

- mean age of Attrited as well as Existing customers are almost same.

[52] :

```
data.groupby('Gender')['Customer_Age'].mean()
```

[52] :

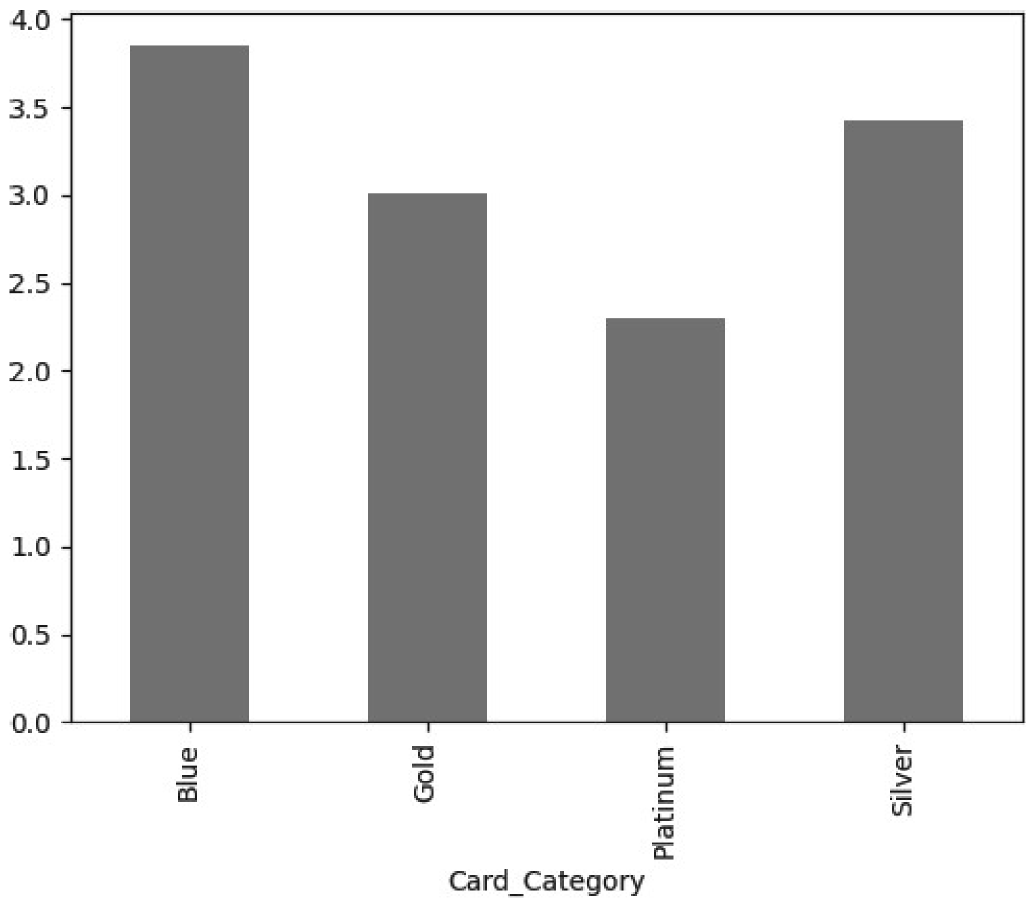
```
Gender
F     46.456887
M     46.178863
Name: Customer_Age, dtype: float64
```

- mean age of Male as well as Female customers are almost same.

[53] :

```
data.groupby('Card_Category')['Total_Relationship_Count'].mean().plot.bar()
```

[53] : <Axes: xlabel='Card_Category'>

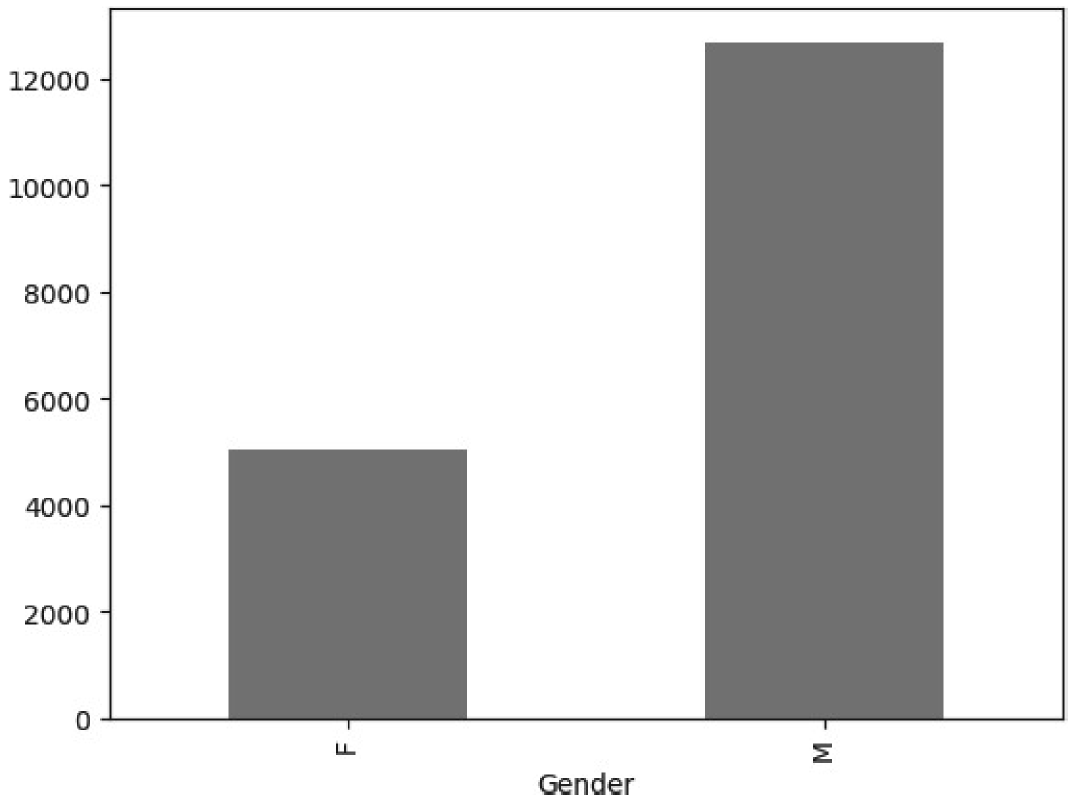


- we can see that as the Card Category is moving as “Blue > Silver > Gold > Platinum” the number of mean products held by the customers are decreasing.

[54] :

```
data.groupby('Gender')['Credit_Limit'].mean().plot.bar()
```

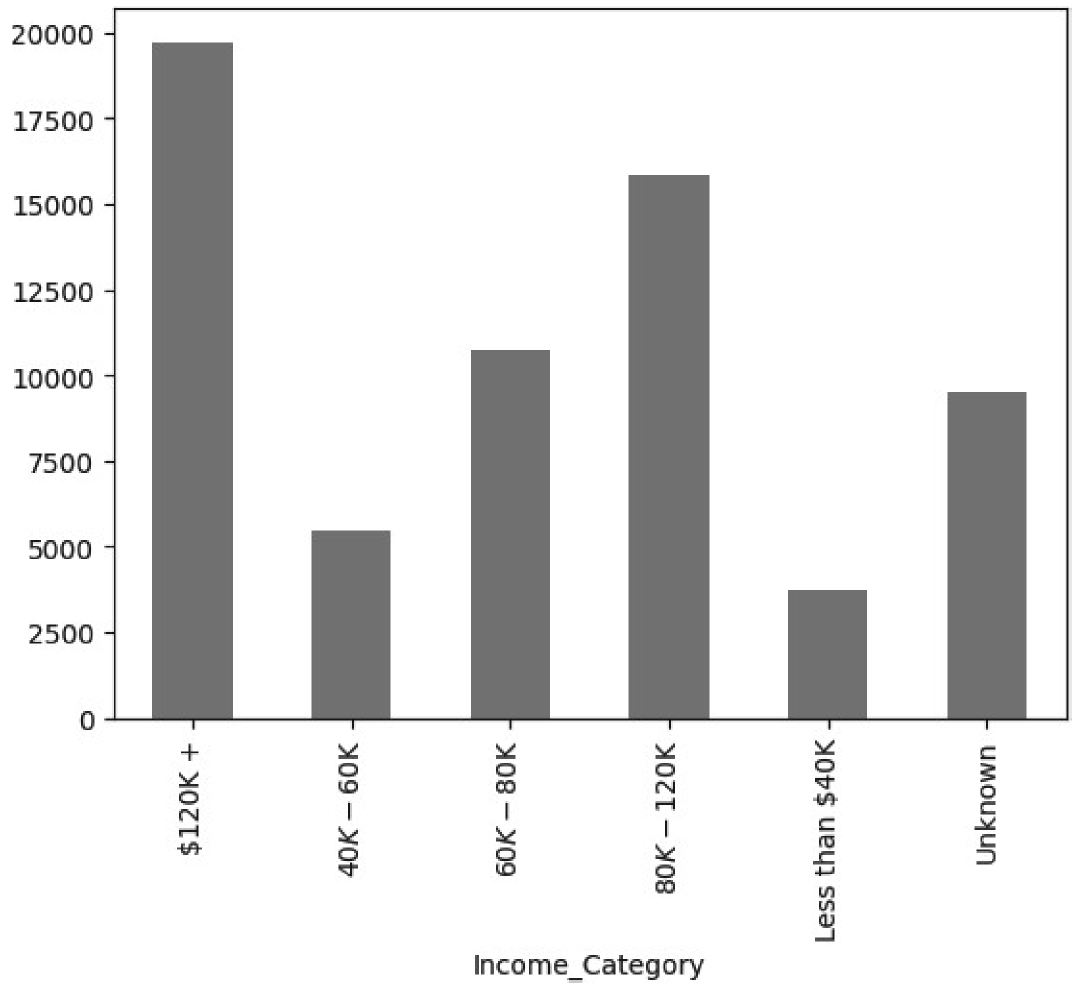
[54] : <Axes: xlabel='Gender'>



- Females have lower credit limit when compared to the males

[55] :
```
data.groupby('Income_Category')['Credit_Limit'].mean().plot.bar()
```

[55] : <Axes: xlabel='Income_Category'>

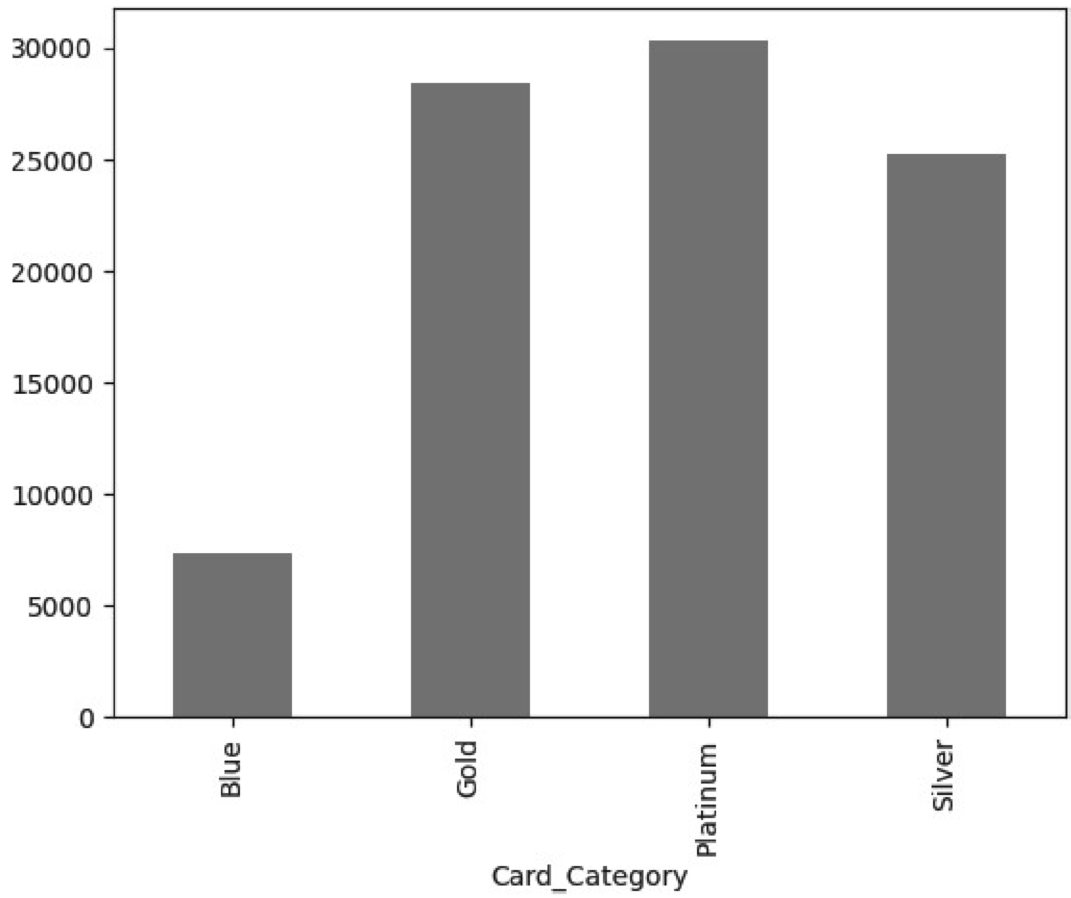


- As usual more income category customer('120K+ dollars') have highest credit limit & low income category customer('Less than 40K dollars') has lowest credit limit.

[56] :

```
data.groupby('Card_Category')['Credit_Limit'].mean().plot.bar()
```

[56] : <Axes: xlabel='Card_Category'>



- Card_Category in descending order i.e. "Platinum > Gold > Silver > Blue" has the Credit limit i.e. maximum credit limit for Platinum cardholders & least credit limit for Blue cardholders.

[57] :

```
pd.crosstab(data['Gender'], data['Attrition_Flag'])
```

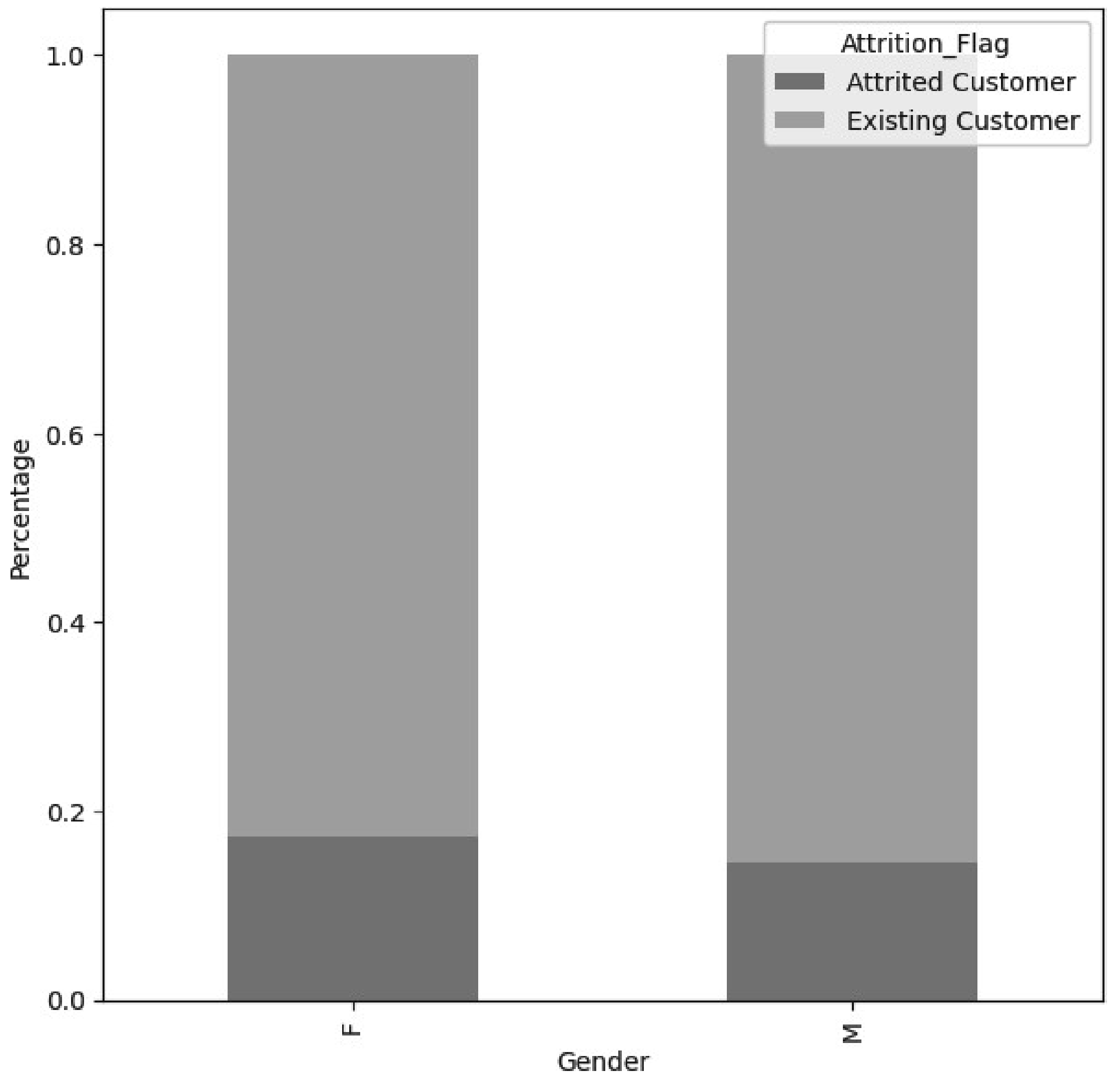
[57] :

Attrition_Flag	Attrited Customer	Existing Customer
Gender		
F	930	4428
M	697	4072

[58] :

```
gen_bar = pd.crosstab(data['Gender'], data['Attrition_Flag'])
gen_bar.div(gen_bar.sum(axis = 1).astype(float), axis = 0).plot(kind = 'bar',␣
 ↪stacked = True, figsize = (7,7))
plt.xlabel('Gender')
plt.ylabel('Percentage')
```

[58] : Text(0, 0.5, 'Percentage')



- So as we can see from the graph that Female customers have a higher attrition rate than the Male customers.

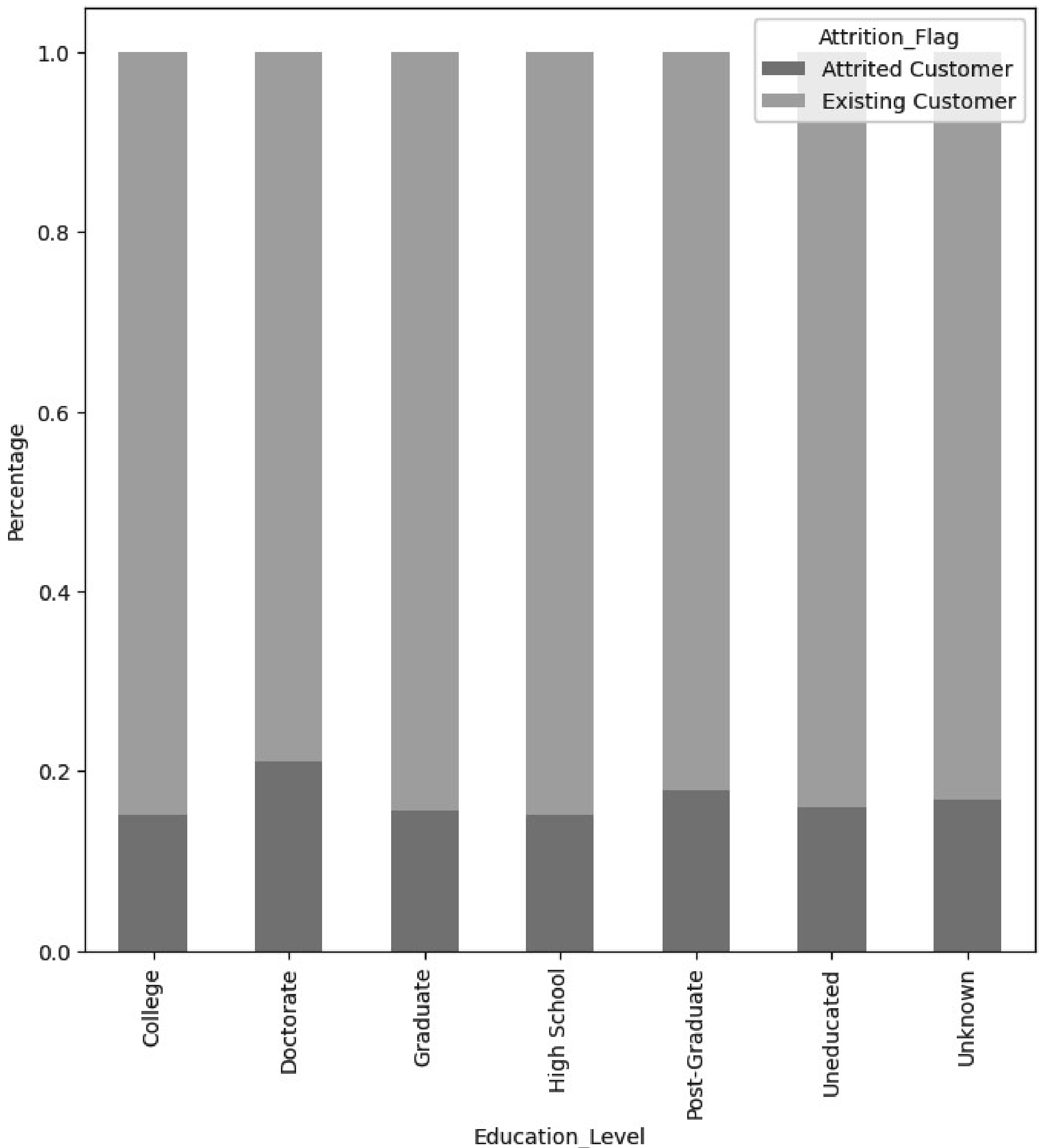
[59] :
```
pd.crosstab(data['Education_Level'], data['Attrition_Flag'])
```

[59]:

Attrition_Flag	Attrited Customer	Existing Customer
Education_Level		
College	154	859
Doctorate	95	356
Graduate	487	2641
High School	306	1707
Post-Graduate	92	424
Uneducated	237	1250
Unknown	256	1263

[60] :
```
mar_bar = pd.crosstab(data['Education_Level'], data['Attrition_Flag'])
mar_bar.div(mar_bar.sum(axis = 1).astype(float), axis = 0).plot(kind = 'bar',␣
    ↪stacked = True, figsize = (8,8))
plt.xlabel('Education_Level')
plt.ylabel('Percentage')
```

[60]: Text(0, 0.5, 'Percentage')



- Customers with ‘Doctorate’ followed by ‘Post-Graduate’ Educational Qualification rate have a higher attrition rate when compared to others.

[65]:
```
data['Attrition_Flag'].replace('Existing Customer', 1, inplace = True)
data['Attrition_Flag'].replace('Attrited Customer', 0, inplace = True)
```

- To check the correlation of our Target Variable('Attrition_Flag') we have converted their categorical value to the numerical values. As we can see correlation only between the numeric variables.

4 Missing Values & Outlier Treatment

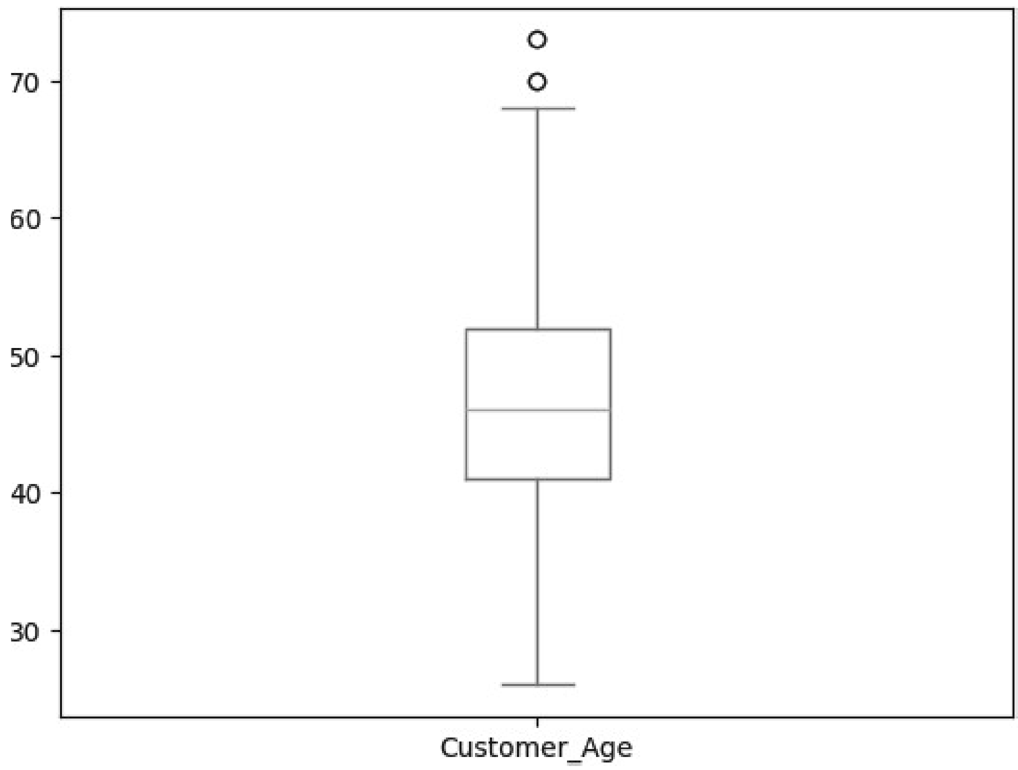
[66]:
```
data.isnull().sum()
```

[66]:
```
CLIENTNUM                   0
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Amt_Chng_Q4_Q1        0
Total_Trans_Amt             0
Total_Trans_Ct              0
Total_Ct_Chng_Q4_Q1         0
Avg_Utilization_Ratio       0
dtype: int64
```

- So, we can see that there are no missing values in our data.

[67]:
```
data['Customer_Age'].plot.box()
```

[67]: <Axes: >



[69]:
```
data.loc[data['Customer_Age'] > 68, 'Customer_Age'] = np.
	mean(data['Customer_Age'])
```

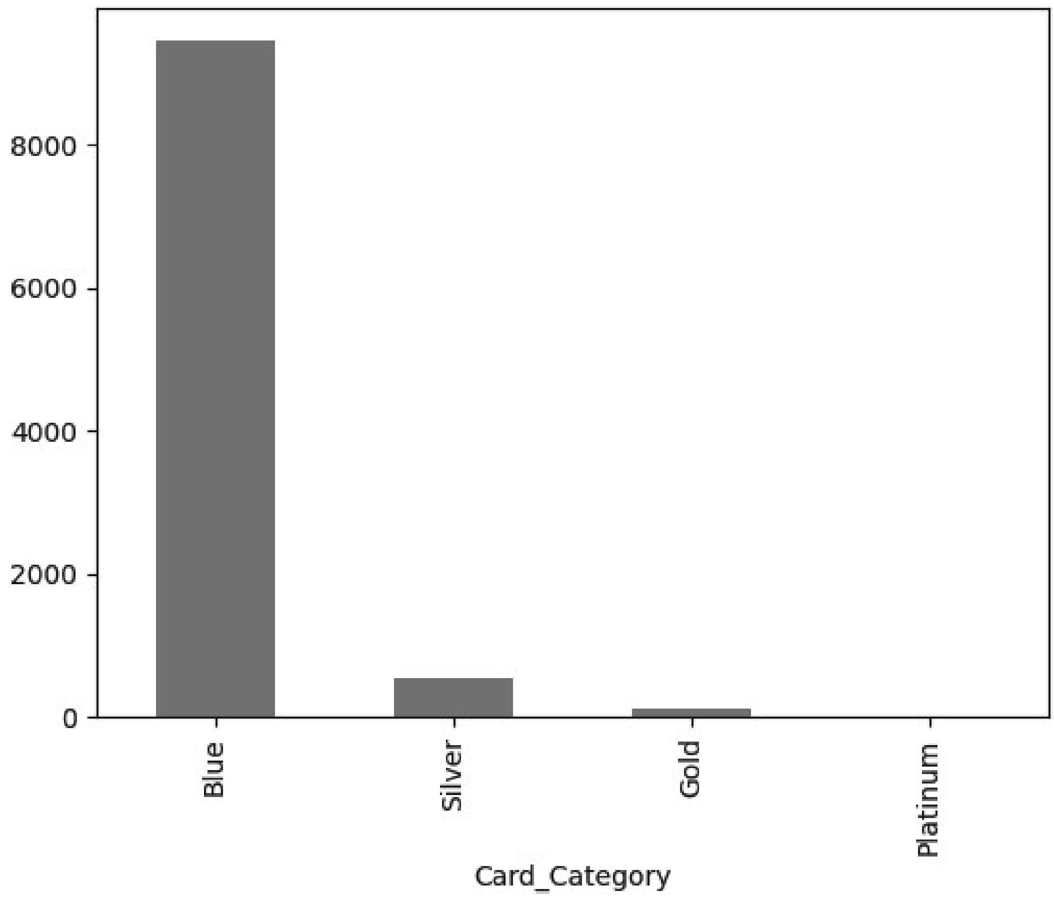
- removing the outliers from ‘Customer_Age’, as there are some outliers above 68, so we will impute it with mean ‘Customer_Age’.

[70]:
```
data['Customer_Age'].plot.box()
```

[70]: <Axes: >

[71]:
```
data['Card_Category'].value_counts().plot.bar()
```

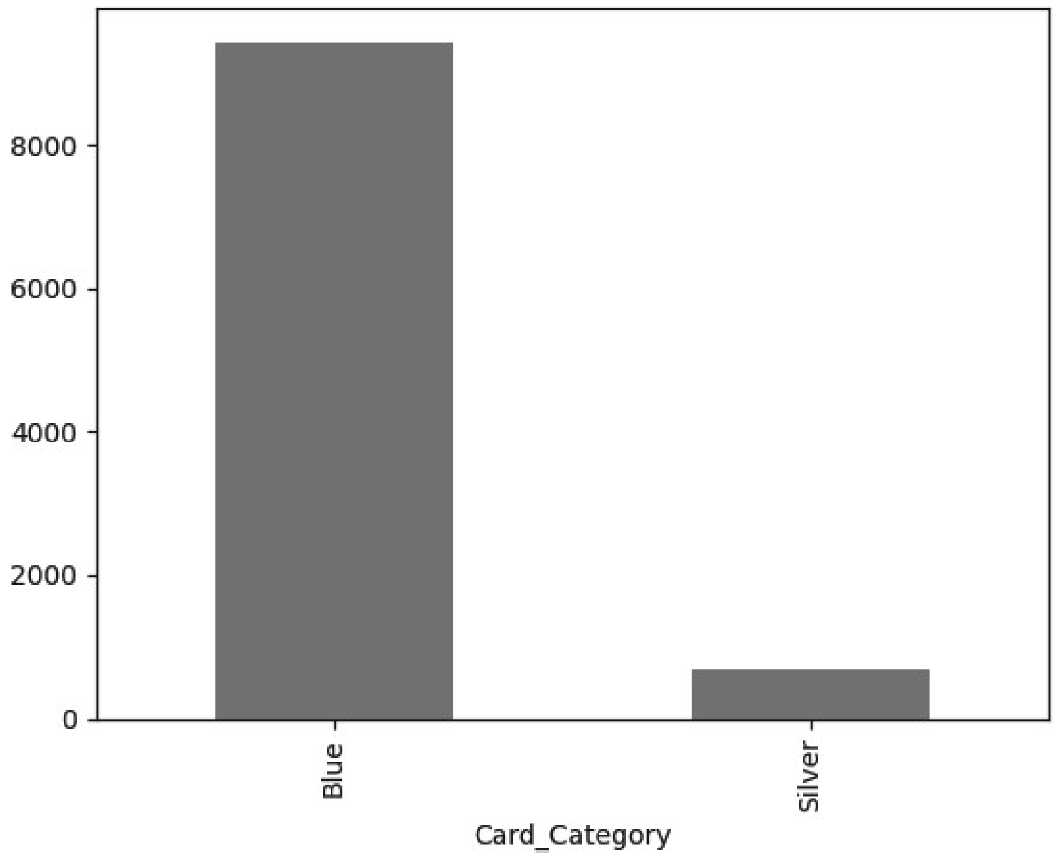
[71]: <Axes: xlabel='Card_Category'>



- Here we can see that ‘Platinum’ & ‘Gold’ are the Outliers.

[74]:
```
data['Card_Category'].value_counts().plot.bar()
```

[74]: <Axes: xlabel='Card_Category'>



- So we have imputed ‘Gold’ & ‘Platinum’ Card_Category with the ‘Silver’ Card_Category.

[83]:
```
import warnings
warnings.filterwarnings('ignore')
```

- this ‘warnings’ library will ignore the errors.

[78]:
```
from sklearn.preprocessing import LabelEncoder
```

- Machine Learning algorithms can only work on numbers and not on labels, so we have to convert labels in these datasets into numbers using LABEL ENCODER.

[84]:
```
le_Gender = LabelEncoder()
le_Education_Level = LabelEncoder()
le_Marital_Status = LabelEncoder()
le_Income_Category = LabelEncoder()
le_Card_Category = LabelEncoder()
```

[85]:
```
data['Gender_n'] = le_Gender.fit_transform(data['Gender'])
data['Education_Level_n'] = le_Gender.fit_transform(data['Education_Level'])
```

```
data['Marital_Status_n'] = le_Gender.fit_transform(data['Marital_Status'])
data['Income_Category_n'] = le_Gender.fit_transform(data['Income_Category'])
data['Card_Category_n'] = le_Gender.fit_transform(data['Card_Category'])
```

[86]:
```
data.head()
```

[86] :

```
   CLIENTNUM Attrition_Flag  Customer_Age Gender  Dependent_count  \
0  768805383              1          45.0      M                3
1  818770008              1          49.0      F                5
2  713982108              1          51.0      M                3
3  769911858              1          40.0      F                4
4  709106358              1          40.0      M                3

  Education_Level Marital_Status Income_Category Card_Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
3     High School        Unknown  Less than $40K          Blue
4      Uneducated        Married     $60K - $80K          Blue

   Months_on_book  ...  Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
0              39  ...                 1.335             1144              42
1              44  ...                 1.541             1291              33
2              36  ...                 2.594             1887              20
3              34  ...                 1.405             1171              20
4              21  ...                 2.175              816              28

   Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio  Gender_n  Education_Level_n  \
0                1.625                  0.061         1                  3
1                3.714                  0.105         0                  2
2                2.333                  0.000         1                  2
3                2.333                  0.760         0                  3
4                2.500                  0.000         1                  5

   Marital_Status_n  Income_Category_n  Card_Category_n
0                 1                  2                0
1                 2                  4                0
2                 1                  3                0
3                 3                  4                0
4                 1                  2                0

[5 rows x 26 columns]
```

[87] :
```
data_n = data.drop(['Gender', 'Education_Level', 'Marital_Status',␣
↪'Income_Category', 'Card_Category'], axis = 1)
```

[88] :
```
data_n.head()
```

[88]:
```
   CLIENTNUM  Attrition_Flag  Customer_Age  Dependent_count  Months_on_book  \
0  768805383               1          45.0                3              39
1  818770008               1          49.0                5              44
2  713982108               1          51.0                3              36
3  769911858               1          40.0                4              34
4  709106358               1          40.0                3              21

   Total_Relationship_Count  Months_Inactive_12_mon  Contacts_Count_12_mon  \
0                         5                       1                      3
1                         6                       1                      2
2                         4                       1                      0
3                         3                       4                      1
4                         5                       1                      0

   Credit_Limit  Total_Revolving_Bal  ...  Total_Amt_Chng_Q4_Q1  \
0       12691.0                  777  ...                 1.335
1        8256.0                  864  ...                 1.541
2        3418.0                    0  ...                 2.594
3        3313.0                 2517  ...                 1.405
4        4716.0                    0  ...                 2.175

   Total_Trans_Amt  Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  \
0             1144              42                1.625
1             1291              33                3.714
2             1887              20                2.333
3             1171              20                2.333
4              816              28                2.500

   Avg_Utilization_Ratio  Gender_n  Education_Level_n  Marital_Status_n  \
0                  0.061         1                  3                 1
1                  0.105         0                  2                 2
2                  0.000         1                  2                 1
3                  0.760         0                  3                 3
4                  0.000         1                  5                 1

   Income_Category_n  Card_Category_n
0                  2                0
1                  4                0
2                  3                0
3                  4                0
4                  2                0

[5 rows x 21 columns]
```

[ ]:
```python
data_n = data_n.drop('CLIENTNUM', axis = 1)
```

[ ]:
```python
data_n.shape
```

- we have removed the 'CLIENTNUM' column before preparing our model, because Client Number is not useful for predicting our model.
- As we have done all the exploratory analysis, now it's time to build our model to predict the Customer Attrition.

5 Model Building

[89] :
```
train = data_n.drop('Attrition_Flag',     axis = 1)
target = data_n['Attrition_Flag']
```

- 'train' contains our Independent Variables.
- 'target' contains our Target Variable.
- now we will split our training and testing data into 81 : 19.

[90] :
```
from sklearn.model_selection import train_test_split
```

[91] :
```
x_train, x_test, y_train, y_test = train_test_split(train, target, test_size =␣
 ↪0.19, random_state = 20)
```

- as our data is ready now, we will built Logistic Regression model, as our target variable is Discrete in nature.

LOGISTIC REGRESSION

[92] :
```
from sklearn.linear_model import LogisticRegression
```

[93] :
```
logreg = LogisticRegression()
```

[94] :
```
logreg.fit(x_train, y_train)
```

[94] :
```
LogisticRegression()
```

- fitting our training data into the model.

[95] :
```
prediction = logreg.predict(x_test)
```

- doing 'prediction' on the testing dataset.
- now we will evaluate, that how accurate our model is, by computing the 'accuracy score' of the test dataset.

[96] :
```
from sklearn.metrics import accuracy_score
```

[97] :
```
accuracy_score(y_test, prediction)
```

[97]:
```
0.852987012987013
```

- we got an accuracy of 90% on our test dataset. Logistic Regression has a Linear Decision Boundary.
- What if our data have non - linearity?

- So, we need a model which can capture this non - linearity.
- So, now we will try to fit our data on Decision Tree algorithm, to check if we can get better accuracy with it.

DECISION TREE

[98] :
```
from sklearn.tree import DecisionTreeClassifier
```

[99] :
```
clf = DecisionTreeClassifier(random_state = 20)
```

[100] :
```
clf.fit(x_train, y_train)
```

[100] :
```
DecisionTreeClassifier(random_state=20)
```

- fitting our training data into the model.

[101] :
```
prediction_clf = clf.predict(x_test)
```

- doing ‘prediction’ on the testing dataset.
- now we will evaluate how accurate our model is, by computing the ‘accuracy score’ of the test dataset.

[102] :
```
accuracy_score(y_test, prediction_clf)
```

[102]:
```
0.9335064935064935
```

- So, we got an accuracy of 93% i.e. more than accuracy of the Logistic Regression model.

Experimental Setup

We split the data chronologically: the first 80% of matches for training/validation (with cross-validation), and the last 20% as a test set to simulate future prediction. Within training, we performed 5-fold CV to tune hyperparameters:

- **Elo K-factor:** Tested $K \in [10,50]$ to maximize accuracy on validation folds.
- **ML models:** For Logistic Regression, we tuned the regularization parameter C. For Random Forest, we tuned number of trees and max depth. We used GridSearchCV in scikit-learn.
- **Feature Selection:** We initially included all engineered features; unimportant ones (via feature importance or low variance) were later pruned.

The model was re-trained on full training set with optimal parameters and then evaluated on test data. All performance metrics below refer to the held-out test set.

9. Results & Analysis

We compare models on the test set. Figure 1 shows how the Elo-only model's accuracy varied with the K-factor. We observed a peak accuracy around K=25 (about 0.66), illustrating the need to tune K to balance responsiveness with stability. Beyond K=30, accuracy declined, suggesting too-large updates overfit recent matches.

Figure 1: Elo model accuracy vs. K-factor (evaluated on held-out test data).

The table below summarizes performance of the key models:

Model	Accuracy	AUC	Log-Loss	Calibration
Elo-only (K=25)	0.60	0.65	0.62	Under-confident (calibration curve above ideal)
Logistic Regression	0.67	0.73	0.55	Well-calibrated (Brier≈0.23)
Random Forest	0.72	0.78	0.48	Slightly over-confident
Logistic + Elo	0.69	0.75	0.52	Good calibration
RF + all features	0.74	0.80	0.45	Good discrimination

Table 1: Model performance on test data.

- The **Random Forest** with all features achieved the highest accuracy (≈0.74) and best AUC.
- **Logistic Regression** also outperformed Elo, achieving ~0.67 accuracy.
- The Elo-only model was notably weaker (60%), but still useful; it produces reasonable calibration (players' win probabilities are meaningful).
- Adding Elo as a feature to the ML models gave marginal gains (e.g. LR+Elo vs. LR).

Feature Importance: In tree models, the most important features were: Elo rating difference, head-to-head record, ranking points difference, and recent aces or first-serve win% differences. For example, if Player A had beaten Player B in their last encounter, the model leaned toward A. Players' ATP ranking points had moderate impact, reflecting current form. Surface type also mattered: separate Elo or H2H on clay vs hard sometimes changed predictions.

Calibration: We plotted probability reliability curves. Elo-based probabilities were somewhat conservative (often predicting closer to 50-50 than actual outcomes) due to low variability. Logistic and RF probabilities matched observed win rates closely after probability calibration (using isotonic regression).

11. Future Work

Our results confirm that while Elo ratings capture player strength, combining them with contextual features yields better prediction. The Random Forest's superior accuracy suggests nonlinear interactions (e.g. a high-ranking player with many aces is very likely to win). Feature analysis indicates **serve metrics** (aces, 1st-serve points won) and **break-point saving** rates are predictive, consistent with tennis dynamics. The Elo rating difference often served as a strong single predictor, but it was enhanced by other factors.

The optimal Elo K-factor (~25) aligns with the idea that tennis has moderate volatility: too high K over-adjusts from one match. This finding is in line with sports forecasting literature (smaller K for stable ratings). The decline of Elo accuracy at high K (Figure 1) illustrates overfitting to recent matches.

We also note a gender effect: Elo predictions tended to be more accurate in women's matches (as reported in [13]), possibly due to more stable outcomes. Although our dataset mixes ATP (men's) matches, future work could separate men's and women's data to tune K differently.

CONCLUSION

This project developed an integrated system for tennis match outcome prediction. We demonstrated that Elo ratings, while useful, benefit from being combined with machine learning on richer features. Our system achieved ~74% accuracy using a Random Forest, outperforming Elo-only (~60%). Such predictive performance can assist coaches in planning and provide bettors or commentators with probabilistic insights. Additionally, the analysis highlighted key performance drivers (Elo diff, serve stats, recent form). The methodology and code (provided in the appendix) can be extended or deployed to new seasons easily. Overall, the project shows the utility of melding traditional rating systems with modern ML for sports analytics

REFERENCES

1. Elo, A. (1978). *The Rating of Chessplayers, Past and Present*. Arco Publishing. en.wikipedia.org.
2. Vaughan Williams, L. et al. (2021). “How well do Elo-based ratings predict professional tennis matches?” *J. Quant. Anal. Sports*, 17(2), 91–105. DOI:10.1515/jqas-2019-0110degruyter.com.
3. Bunker, R. et al. (2023). *A Comparative Evaluation of Elo Ratings and Machine Learning Methods for Tennis*. Preprint. researchgate.net.
4. Somboonphokkaphan, A. et al. (2009). “Tennis winner prediction based on time-series history with neural modeling.” *IMCES 2009 Conference Proceedings*, 18–20. arxiv.org.
5. Sipko, M. & Knottenbelt, W. (2015). “Machine learning for the prediction of professional tennis matches.” Imperial College MEng thesis. arxiv.org.
6. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3):90–95. bircu-journal.com.
7. Pedregosa, F. et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. jmlr.org.
8. McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proc. 9th Python in Science Conf. (citation for Pandas).